CHAPTER 16

Speech Synthesis

The speech synthesis module of a TTS system is the component that generates the waveform. The input of traditional speech synthesis systems is a phonetic transcription with its associated prosody. The input can also include the original text with tags, as this may help in producing higher-quality speech.

Speech synthesis systems can be classified into three types according to the model used in the speech generation. *Articulatory synthesis*, described in Section 16.2.4, uses a physical model of speech production that includes all the articulators described in Chapter 2. *Formant synthesis* uses a source-filter model, where the filter is characterized by slowly varying formant frequencies; it is the subject of Section 16.2. *Concatenative synthesis* generates speech by concatenating speech segments and is described in Section 16.3. To allow more flexibility in concatenative synthesis, a number of prosody modification techniques are described in Sections 16.4 and 16.5. Finally, a guide to evaluating speech synthesis systems is included in Section 16.6.

Speech synthesis systems can also be classified according to the degree of manual intervention in the system design into *synthesis by rule* and *data-driven synthesis*. In the former, a set of manually derived rules is used to drive a synthesizer, and in the latter the synthesizer's parameters are obtained automatically from real speech data. Concatenative systems are, thus, data driven. Formant synthesizers have traditionally used synthesis by rule, since the evolution of formants in a formant synthesizer has been done with hand-derived rules. Nonetheless, formant transitions can also be trained from data, as we show in Section 16.2.3.

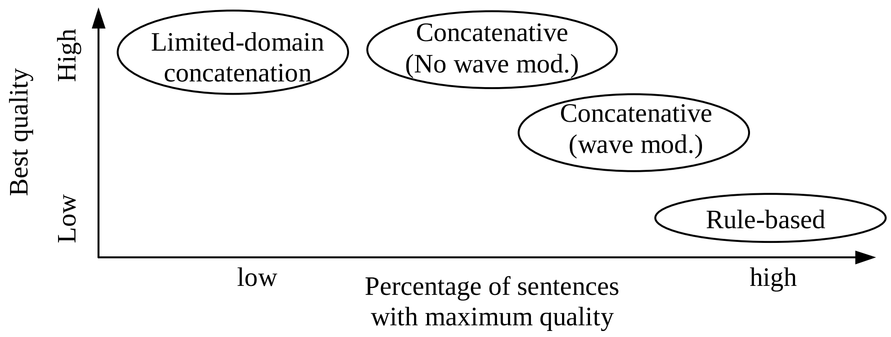

Figure 16.1 Quality and task-independence in speech synthesis approaches.

16.1. ATTRIBUTES OF SPEECH SYNTHESIS

The most important attribute of a speech synthesis system is the quality of its output speech. It is often the case that a single system can sound beautiful on one sentence and terrible on the next. For that reason we need to consider the quality of the best sentences and the percentage of sentences for which such quality is achieved. This tradeoff is illustrated in Figure 16.1, where we compare four different families of speech generation approaches:

- *Limited-domain waveform concatenation*. For a given limited domain, this approach can generate very high quality speech with only a small number of recorded segments. Such an approach, used in most interactive voice response systems, cannot synthesize arbitrary text. Many concept-to-speech systems, described in Chapter 17, use this approach.
- *Concatenative synthesis with no waveform modification*. Unlike the previous approach, these systems can synthesize speech from arbitrary text. They can achieve good quality on a large set of sentences, but the quality can be mediocre for many other sentences where poor concatenations take place.
- *Concatenative systems with waveform modification*. These systems have more flexibility in selecting the speech segments to concatenate because the waveforms can be modified to allow for a better prosody match. This means that the number of sentences with mediocre quality is lower than in the case where no prosody modification is allowed. On the other hand, replacing natural with syn-

thetic prosody can hurt the overall quality. In addition, the prosody modification process also degrades the overall quality.

- *Rule-based systems*. Such systems tend to sound uniformly across different sentences, albeit with quality lower than the best quality obtained in the systems above.

Best-quality and quality variability are possibly two of the most important attributes of a speech synthesis system, but not the only ones. Measuring quality, difficult to do in an objective way, is the main subject of Section 16.6. Other attributes of a speech synthesis system include:

- *Delay*. The time it takes for the synthesizer to start speaking is important for interactive applications and should be less than 200 ms. This delay is composed of the algorithmic delays of the front end and of the speech synthesis module, as well as the computation involved.
- *Memory resources*. Rule-based synthesizers require, on the average, less than 200 KB, so they are a widely used option whenever memory is at a premium. However, required RAM can be an issue for concatenative systems, some of which may require over 100 MB of storage.
- *CPU resources*. With current CPUs, processing time is typically not an issue, unless many channels need to run in the same CPU. Nonetheless, some concatenative synthesizers may require a large amount of computation when searching for the optimal sequence.
- *Variable speed*. Some applications may require the speech synthesis module to generate variable speed, particularly fast speech. This is widely used by blind people who need TTS systems to obtain their information and can accept fast speech because of the increased throughput. Fast speech is also useful when skimming material. Concatenative systems that do not modify the waveform cannot achieve variable speed control, unless a large number of segments are recorded at different speeds.
- *Pitch control*. Some spoken language systems require the output speech to have a specific pitch. This is the case if you want to generate voice for a song. Again, concatenative systems that do not modify the waveform cannot do this, unless a large number of speech segments are recorded at different pitch.
- *Voice characteristics*. Other spoken language systems require specific voices, such as that of a robot, that cannot be recorded naturally, or some, such as monotones, that are tedious to record. Since rule-based systems are so flexible, they are able to do many such modifications.

The approaches described in this chapter assume as input a phonetic string, durations, a pitch contour, and possibly volume. Pauses are signaled by the default phoneme SIL with its corresponding duration. If the parsed text is available, it is possible to do even better in a concatenative system by conducting a matching with all the available information.

16.2. Formant Speech Synthesis

As discussed in Chapter 6, we can synthesize a stationary vowel by passing a glottal periodic waveform through a filter with the formant frequencies of the vocal tract. For the case of unvoiced speech we can use random noise as the source instead. In practice, speech signals are not stationary, and we thus need to change the pitch of the glottal source and the formant frequencies over time. The so-called *synthesis-by-rule* refers to a set of rules on how to modify the pitch, formant frequencies, and other parameters from one sound to another while maintaining continuity present in physical systems like the human production system. Such a system is described in the block diagram of Figure 16.2.

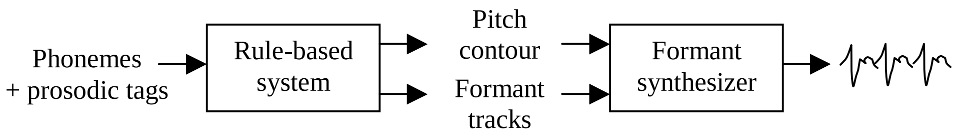

Figure 16.2 Block diagram of a synthesis-by-rule system. Pitch and formants are listed as the only parameters of the synthesizer for convenience. In practice, such system has about 40 parameters.

In Section 16.2.1 we describe the second block of Figure 16.2, the formant synthesizer that generates a waveform from a set of parameters. In Section 16.2.2 we describe the first block of Figure 16.2, the set of rules that can generate such parameters. This approach was the one followed by Dennis Klatt and his colleagues [4, 30]. A data-driven approach to this first block is studied in Section 16.2.3. Finally, articulatory synthesis is the topic of Section 16.2.4.

16.2.1. Waveform Generation from Formant Values

To be able to synthesize speech by rule, a simple model for the filter is needed. Most rule-based synthesizers use the so-called *formant synthesis*, which is derived from models of speech production. The model explicitly represents a number of formant resonances (from 2 to 6). A formant resonance can be implemented (see Chapter 6) with a second-order IIR filter

$$H_i(z) = \frac{1}{1 - 2e^{-\pi b_i}\cos(2\pi f_i)z^{-1} + e^{-2\pi b_i}z^{-2}} \tag{16.1}$$

with $f_i = F_i / F_s$ and $b_i = B_i / F_s$, where F_i, B_i, and F_s are the formant's center frequency, formant's bandwidth, and sampling frequency, respectively, all in Hz. A filter with several resonances can be constructed by cascading several such second-order sections (*cascade model*) or by adding several such sections together (*parallel model*). Formant synthesizers typically use the parallel model to synthesize fricatives and stops and the cascade model for all voiced sounds.

Unlike the cascade model, the parallel model requires gains to be specified for each second-order section, which often are chosen proportional to the formant's frequency and inversely proportional to the formant's bandwidth. The cascade model results in an all-pole filter, whereas the parallel model has zeros in addition to poles. Such a combination is shown in Figure 16.3, where R1 through R6 are the resonances 1 to 6 and each one represents a second-order IIR filter like that in Eq. (16.1). RNP represents the nasal resonance, and RNZ is an FIR filter with the nasal zero. A1 through AB are the gains for each filter, used for the parallel model. Switch SW controls whether the cascade model or parallel model is used.

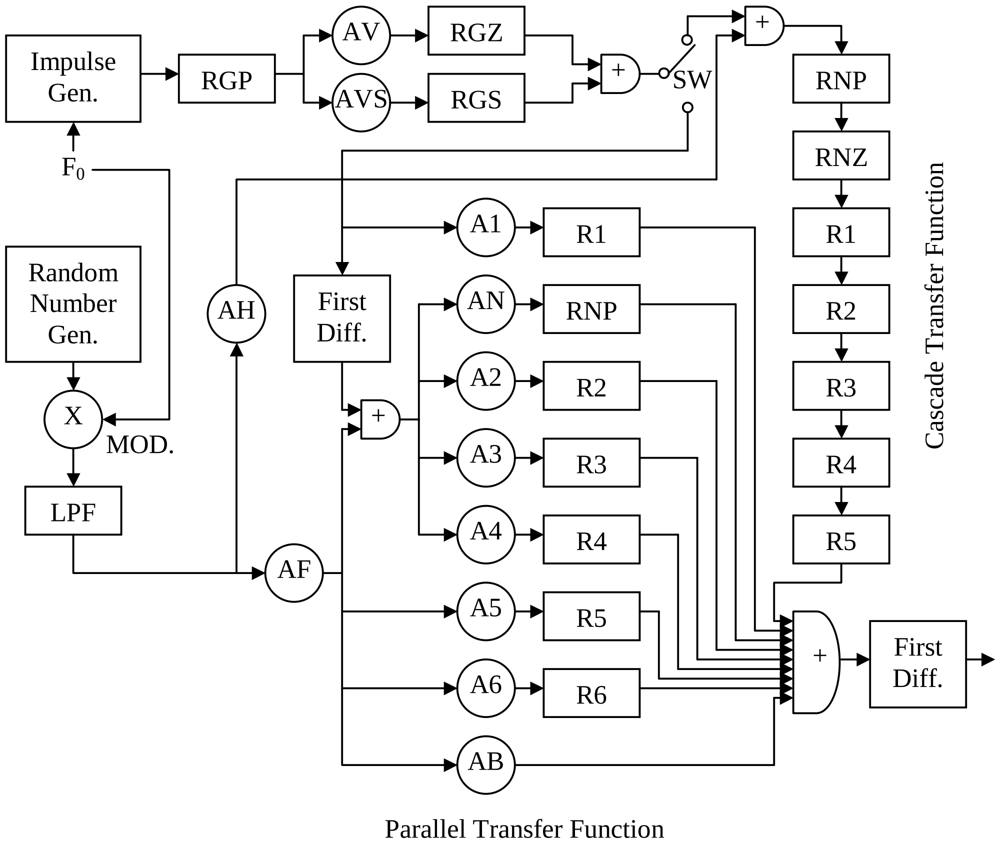

Figure 16.3 Block diagram of the Klatt formant synthesizer (after Klatt [4]).

For voiced sounds the excitation model consists of an impulse train driving a low-pass filter RGP and then a bandpass filter created by the parallel of RGZ and RGS. For unvoiced sounds the excitation consists of white noise driving a low-pass filter LPF. The excitation for voiced fricatives is a combination of the two sources. In practice, this mixed excitation is used for all voiced sounds to add some breathiness. Klatt [30] showed that this model could reproduce quite faithfully a natural recording if the parameters had been manually selected.

Table 16.1 Parameter values for Klatt's cascade/parallel formant synthesizer with the parameter symbol, full name, minimum, maximum, and typical values (after Allen [4]).

N	Symbol	Name	Min	Max	Typ
1	AV	Amplitude of voicing (dB)	0	80	0
2	AF	Amplitude of frication (dB)	0	80	0
3	AH	Amplitude of aspiration (dB)	0	80	0
4	AVS	Amplitude of sinusoidal voicing (dB)	0	80	0
5	F0	Fundamental frequency (Hz)	0	500	0
6	F1	First formant frequency (Hz)	150	900	500
7	F2	Second formant frequency (Hz)	500	2500	1500
8	F3	Third formant frequency (Hz)	130	3500	2500
9	F4	Fourth formant frequency (Hz)	250	4500	3500
10	FNZ	Nasal zero frequency (Hz)	200	700	250
11	AN	Nasal formant amplitude (Hz)	0	80	0
12	A1	First formant amplitude (Hz)	0	80	0
13	A2	Second formant amplitude (Hz)	0	0	0
14	A3	Third formant amplitude (Hz)	0	80	0
15	A4	Fourth formant amplitude (Hz)	0	80	0
16	A5	Fifth formant amplitude (Hz)	0	80	0
17	A6	Sixth formant amplitude (Hz)	0	80	0
18	AB	Bypass path amplitude (Hz)	0	80	0
19	B1	First formant bandwidth (Hz)	40	500	50
20	B2	Second formant bandwidth (Hz)	40	500	70
21	B3	Third formant bandwidth (Hz)	40	500	110
22	SW	Cascade/parallel switch	0	1	0
23	FGP	Glottal resonator 1 frequency (Hz)	0	600	0
24	BGP	Glottal resonator 1 bandwidth (Hz)	100	2000	100
25	FGZ	Glottal zero frequency (Hz)	0	500	1500
26	BGZ	Glottal zero bandwidth (Hz)	100	9000	6000
27	B4	Fourth formant bandwidth (Hz)	100	500	250
28	F5	Fifth formant frequency (Hz)	350	4900	3850
29	B5	Fifth formant bandwidth (Hz)	150	700	200
30	F6	Sixth formant frequency (Hz)	400	4999	4900
31	B6	Sixth formant bandwidth (Hz)	200	2000	100
32	FNP	Nasal pole frequency (Hz)	200	500	250
33	BNP	Nasal pole bandwidth (Hz)	50	500	100
34	BNZ	Nasal zero bandwidth (Hz)	50	500	100
35	BGS	Glottal resonator 2 bandwidth (Hz)	100	1000	200
36	SR	Sampling rate (Hz)	500	2000	1000
37	NWS	Number of waveform samples per chunk	1	200	50
38	G0	Overall gain control (dB)	0	80	48
39	NFC	Number of cascaded formants	4	6	5

The parameter names and their minimum and maximum values are listed in Table 16.1, where the switch SW can be in voiced (V) or consonant (C) mode. In Figure 16.3, R2 is the resonator corresponding to the second formant, whose center frequency F2 and bandwidth B2 are given in Table 16.1. In addition to the six resonances associated to the six formants, there are other resonances: RGP, RGZ, RGS, RNP, and RNZ. Other source models are also possible [43].

16.2.2. Formant Generation by Rule

As described in Chapter 2, formants are one of the main features of vowels. Because of the physical limitations of the vocal tract, formants do not change abruptly with time. Rule-based formant synthesizers enforce this by generating continuous values for $f_i[n]$ and $b_i[n]$, typically every 5–10 milliseconds. Continuous values can be implemented through the above structures if a lattice filter is used, because it allows its reflection coefficients to vary at every sample (see Chapter 6). In practice, the values can be fixed within a frame as long as frames are smaller than 5 ms.

Rules on how to generate formant trajectories from a phonetic string are based on the *locus theory* of speech production. The locus theory specifies that formant frequencies within a phoneme tend to reach a stationary value called the *target*. Targets for formant frequencies and bandwidths for a male speaker are shown in Table 16.2 (nonvocalic segments) and Table 16.3 (vocalic segments). This target is reached if either the phoneme is sufficiently long or the previous phoneme's target is close to the current phoneme's target. The maximum slope at which the formants move is dominated by the speed of the articulators, dominated by physical constraints. Since each formant is primarily caused by the position of a given articulator, formants caused by the body of the tongue do not vary as rapidly as formants caused by the tip of the tongue or the lips. Thus, rule-based systems store targets for each phoneme as well as maximum allowable slopes and transition times.

For example, a transition between a vowel and a sonorant can follow the rule shown in Figure 16.4 with a_1 being the target of unit 1 and a_2 the target of unit 2. The values of T_{cb} and T_{cf} are 40 and 80 ms, respectively, and $a_b = a_2 + 0.75(a_1 - a_2)$. The time $T_{cb} + T_{cf}$ specifies how rapid the transition is. While linear interpolation could be used, a_b and the ratio T_{cb} / T_{cf} are sometimes used to further refine the shape of the formant transition.

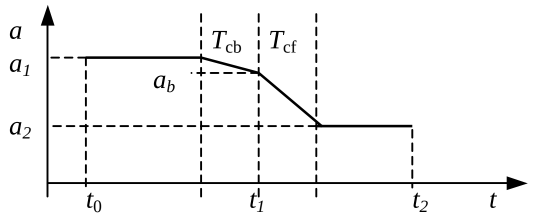

Figure 16.4 Transition between two vowels in a formant synthesizer.

Other rules can allow a discontinuity, for example when a transition out of an unvoiced segment takes place. To improve naturalness, all these parameters can be made dependent on the immediate phonetic context.

Table 16.2 Targets used in the Klatt synthesizers: formant frequencies and bandwidths for nonvocalic segments of a male speaker. Note that in addition to the phoneme set used in Chapter 2, there are several additional phonetic segments here such as *axp, dx, el, em, en, gp, hx, kp, lx, qq, rx, tq, wh*, that allow more control (after Allen [4]).

	F1	F2	F3	B1	B2	B3
axp	430	1500	2500	120	60	120
b	200	900	2100	65	90	125
ch	300	1700	2400	200	110	270
d	200	1400	2700	70	115	180
dh	300	1150	2700	60	95	185
dx	200	1600	2700	120	140	250
el	450	800	2850	65	60	80
em	200	900	2100	120	60	70
en	200	1600	2700	120	70	110
ff	400	1130	2100	225	120	175
g	250	1600	1900	70	145	190
gp	200	1950	2800	120	140	250
h	450	1450	2450	300	160	300
hx	450	1450	2450	200	120	200
j	200	1700	2400	50	110	270
k	350	1600	1900	280	220	250
kp	300	1950	2800	150	140	250
l	330	1050	2800	50	100	280
lx	450	800	2850	65	60	80
m	480	1050	2100	40	175	120
ng	480	1600	2050	160	150	100
n	480	1400	2700	40	300	260
p	300	900	2100	300	190	185
qq	400	1400	2450	120	140	250
r	330	1060	1380	70	100	120
rx	460	1260	1560	60	60	70
sh	400	1650	2400	200	110	280
sil	400	1400	2400	120	140	250
s	400	1400	2700	200	95	220
th	400	1150	2700	225	95	200
tq	200	1400	2700	120	140	250
t	300	1400	2700	300	180	220
v	300	1130	2100	55	95	125
wh	330	600	2100	150	60	60
w	285	610	2150	50	80	60
y	240	2070	3020	40	250	500
zh	300	1650	2400	220	140	250
z	300	1400	2700	70	85	190

Klatt showed that for a given natural utterance, he could manually obtain a sequence of formant tracks $f_i[n]$ and $b_i[n]$, such that the synthesized utterance not only had good quality but also sounded very similar to the original. This shows that the formant synthesizer of Section 16.2.1 appears to be sufficient for generation. On the other hand, when the formant tracks are obtained automatically through rules such as that of Figure 16.4 and Table 16.2 and Table 16.3, the output speech does not sound that natural, and the voice does not resemble the voice of the original recording.

Table 16.3 Targets used in the Klatt synthesizers: formant frequencies and bandwidths for vocalic segments of a male speaker. Note that in addition to the phoneme set used in Chapter 2, there are several additional phonetic segments here such as *axr, exr, ix, ixr, oxr, uxr, yu* that allow more control (after Allen [4]).

	F1	F2	F3	B1	B2	B3
aa	700	1220	2600	130	70	160
ae	620	1660	2430	70	130	300
ah	620	1220	2550	80	50	140
ao	600	990	2570	90	100	80
aw	640	1230	2550	80	70	110
ax	550	1260	2470	80	50	140
axr	680	1170	2380	60	60	110
ay	660	1200	2550	100	120	200
eh	530	1680	2500	60	90	200
er	470	1270	1540	100	60	110
exr	460	1650	2400	60	80	140
ey	480	1720	2520	70	100	200
ih	400	1800	2670	50	100	140
ix	420	1680	2520	50	100	140
ixr	320	1900	2900	70	80	120
iy	310	2200	2960	50	200	400
ow	540	1100	2300	80	70	70
oxr	550	820	2200	60	60	60
oy	550	960	2400	80	120	160
uh	450	1100	2350	80	100	80
uw	350	1250	2200	65	110	140
uxr	360	800	2000	60	60	80
yu	290	1900	2600	70	160	220

Formant synthesis is very flexible because it can generate intelligible speech with relatively few parameters (about 40). The use of context-dependent rules can improve the quality of the synthesizer at the expense of a great deal of manual tuning. The synthesized speech is, by design, smooth, although it may not resemble any given speaker and may not sound very natural.

Because of their flexibility, formant synthesizers can often generate many different voices and effects. While not as flexible, voice effects can also be produced in concatenative speech systems (see Section 16.5.3).

16.2.3. Data-Driven Formant Generation

While, in general, formant synthesis assumes the formant model of Section 16.2.1 driven by parameter values generated by rules, as in Section 16.2.2, data-driven methods to generate the formant values have also been proposed [3]. An HMM running in generation mode emits three formant frequencies and their bandwidths every 10 ms, and these values are used in a cascade formant synthesizer similar to that described in Section 16.2.1. Like the speech recognition counterparts, this HMM has many decision-tree context-dependent triphones and three states per triphone. A Gaussian distribution per state is used in this work. The baseline system uses a six-dimensional vector that includes the first three formant frequencies and their bandwidths. Initially it is assumed that the input to the synthesizer includes, in addition to the duration of each phoneme, the duration of each state. In this case, the maximum likelihood formant track is a sequence of the state means and, therefore, is discontinuous at state boundaries.

The key to obtaining a smooth formant track is to augment the feature vector with the corresponding delta formants and bandwidths (the difference between the feature at time t and that feature at time $t-1$) to complete a twelve-dimensional vector. The maximum likelihood solution now entails solving a tridiagonal set of linear equations (see the discussion on statistical formant tracking in Chapter 6). The resulting formant track is smooth, as it balances formant values that are close to the state means with delta values that are also within the state means. In addition, the synthesized speech resembles that of the donor speaker. More details on the analysis and model training can be found in the formant tracking section of Chapter 6.

16.2.4. Articulatory Synthesis

Articulatory synthesis is another model that has been used to synthesize speech by rule, by using parameters that model the mechanical motions of the articulators and the resulting distributions of volume velocity and sound pressure in the lungs, larynx, and vocal and nasal tracts. Because the human speech production articulators do not have that many degrees of freedom, articulatory models often use as few as 15 parameters to drive a formant synthesizer.

The relationship between articulators and acoustics is many-to-one [17]. For example, a ventriloquist can produce speech sounds with very different articulator positions than those of normal speech. The *speech inversion* problem is therefore ill posed. By using the assumption that the articulators do not change rapidly over time, it is possible, however, to estimate the vocal-tract area from formant frequencies [37]. In [8] the model uses five articulatory parameters: area of lip opening, constriction formed by the tongue blade, opening to the nasal cavities, average glottal area, and rate of active expansion or contraction of the vocal tract volume behind a constriction. These five parameters are augmented with the first four formant frequencies and F0.

Those area parameters can be obtained from real speech through X-rays and magnetic resonance imaging (MRI), though positioning such sensors in the vocal tract alters the way speech is produced (such as the sensors in the lips) and impedes completely natural sounds. The relationship between articulatory parameters and acoustic values has typically been done using a nonlinear mapping such as a neural network or a codebook.

While one day this may be the best way to synthesize speech, the state-of-the-art in articulatory synthesis does not generate speech with quality comparable to that of formant or concatenative systems.

16.3. CONCATENATIVE SPEECH SYNTHESIS

While state-of-the-art synthesis by rule is quite intelligible, it sounds unnatural, because it is very difficult to capture all the nuances of natural speech in a small set of manually derived rules. In *concatenative synthesis*, a speech segment is *synthesized* by simply playing back a waveform with matching phoneme string. An utterance is synthesized by concatenating together several speech fragments. The beauty of this approach is that unlike synthesis-by-rule, it requires neither rules nor manual tuning. Moreover, each segment is completely natural, so we should expect very natural output.

Unfortunately, this is equivalent to assembling an automobile with parts of different colors: each part is very good yet there is a color discontinuity from part to part that makes the whole automobile unacceptable. Speech segments are greatly affected by coarticulation [42], so if we concatenate two speech segments that were not adjacent to each other, there can be spectral or prosodic discontinuities. Spectral discontinuities occur when the formants at the concatenation point do not match. Prosodic discontinuities occur when the pitch at the concatenation point does not match. A listener rates as poor any synthetic speech that contains large discontinuities, even if each segment is very natural.

Thus, when designing a concatenative speech synthesis system we need to address the following issues:

1. What type of speech segment to use? We can use diphones, syllables, phonemes, words, phrases, etc.
2. How to design the acoustic inventory, or set of speech segments, from a set of recordings? This includes excising the speech segments from the set of recordings as well as deciding how many are necessary. This is similar to the training problem in speech recognition.
3. How to select the best string of speech segments from a given library of segments, and given a phonetic string and its prosody? There may be several strings of speech segments that produce the same phonetic string and prosody. This is similar to the search problem in speech recognition.
4. How to alter the prosody of a speech segment to best match the desired output prosody. This is the topic of Section 16.4.

Generally, these concatenative systems suffer from great variability in quality: often they can offer excellent quality in one sentence and terrible quality in the next one. If enough good units are available, a given test utterance can sound almost as good as a recorded utterance. However, if several discontinuities occur, the synthesized utterance can have very poor quality. While synthesizing arbitrary text is still a challenge with these techniques, for restrictive domains this approach can yield excellent quality. We examine all these issues in the following sections.

We define *unit* as an abstract representation of a speech segment, such as its phonetic label, whereas we use *instance* as a speech segment from an utterance that belongs to the same unit. Thus, a system can keep several instances of a given unit to select among them to better reduce the discontinuities at the boundaries. This abstract representation consists of the unit's phonetic transcription at the minimum in such a way that the concatenation of a set of units matches the target phonetic string. In addition to the phonetic string, this representation can often include prosodic information.

Table 16.4 Unit types in English assuming a phone set of 42 phonemes. Longer units produce higher quality at the expense of more storage. The number of units is generally below the absolute maximum in theory*:* i.e., out of the 42^3 = 74,088 possible triphones, only about 30,000 occur in practice.

Unit length	Unit type	#Units	Quality
Short	Phoneme	42	Low
↓	Diphone	~1500	↓
	Triphone	~30K	
	Demisyllable	~2000	
	Syllable	~11K	
	Word	100K–1.5M	
	Phrase	∞	
Long	Sentence	∞	High

16.3.1. Choice of Unit

A number of units that have been used in the field, including context-independent phonemes, diphones, context-dependent phonemes, subphonetic units, syllables, words, and phrases. A compilation of unit types for English is shown in Table 16.4 with their coverage in Figure 16.5.

The issues in choosing appropriate units for synthesis are similar to those in choosing units for speech recognition (described in Chapter 9):

- The unit should lead to *low concatenation distortion*. A simple way of minimizing this distortion is to have fewer concatenations and thus use long units such as words, phrases or even sentences. But since some concatenations are unavoidable, we also want to use units that naturally lead to "small" discontinuities at the concatenation points. For example, it has been observed that spectral discontinuities at vowels are much more noticeable than at fricatives, or that a discontinuity within a syllable is more perceptually noticeable than a discontinuity across syllable boundaries, and similarly for within-word and across-word discontinuities [55]. Having several instances per unit is an alternative to long units that allows the choice of instances with low concatenation distortion.

- The unit should lead to *low prosodic distortion*. While it is not crucial to have units with slightly different prosody than the desired target, replacing a unit with a rising pitch with another with a falling pitch may result in an unnatural sentence. Altering the pitch and or duration of a unit is possible (see Section 16.4) at the expense of additional distortion.
- The unit should be *generalizable*, if unrestricted text-to-speech is required. If we choose words or phrases as our units, we cannot synthesize arbitrary speech from text, because it's almost guaranteed that the text will contain words not in our inventory. As an example, the use of arbitrarily long units in such a way that no concatenation between voiced sounds occurs by cutting at obstruents results in low concatenation distortion but it is shown [47] that over 180,000 such units would be needed to cover 75% of a random corpus. The longer the speech segments are, the more of them we need to be able to synthesize speech from arbitrary text. This generalization property is not needed if closed-domain synthesis is desired.
- The unit should be *trainable*. Our training data should be sufficient to estimate all our units. Since the training data is usually limited, having fewer units leads to better trainability in general. So the use of words, phrases, or sentences may be prohibitive other than for closed-domain synthesis. The other units in Table 16.4 can be considered trainable depending on the limitations on the size of our acoustic inventory.

A practical challenge is how to balance these selection criteria. In this section we compare a number of units and point out their strengths and weaknesses.

16.3.1.1. Context-Independent Phonemes

The most straightforward unit is the phoneme. Having one instance of each phoneme, independent of the neighboring phonetic context, is very generalizable, since it allows us to generate every word/sentence. It is also very trainable and we could have a system that is very compact. For a language with N phonemes, say $N = 42$ for English, we would need only N unit instances. The problem is that using *context-independent* phones results in many audible discontinuities. Such a system is not intelligible.

16.3.1.2. Diphones

A type of subword unit that has been extensively used is the so-called *dyad* or *diphone* [41]. A diphone `S-IH` includes from the middle of the `S` phoneme to the middle of the `IH` phoneme, so diphones are, on the average, one phoneme long. The word `hello` */hh ax l ow/* can be mapped into the diphone sequence: */sil-hh/, /hh-ax/, /ax-l/, /l-ow/, /ow-sil/*. If our language has *N* phonemes, there are potentially N^2 diphones. In practice, many such diphones never occur in the language, so that a smaller number is sufficient. For example, the phonetic alphabet of Chapter 2 has 42 phonemes for English, and only about 1300 diphones are needed. Diphone units were among the first type of unit used in concatenative systems because it yields fairly good quality. While diphones retain the transitional information, there can be large distortions due to the difference in spectra between the stationary parts of two units obtained from different contexts. For example, there is no guarantee that the spectra of */ax-l/* and */l-ow/* will match at the junction point, since */ax-l/* could have a very different right context than */ow/* or */l-ow/* could have a very different left context than */ax/*.

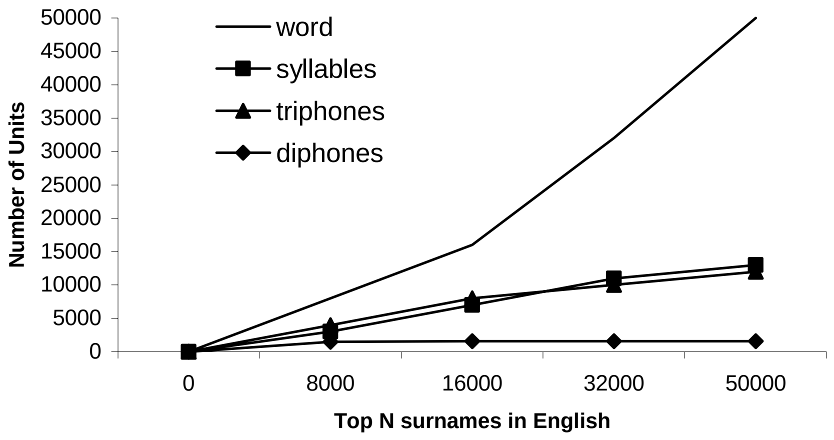

Figure 16.5 Coverage with different number of units displays the number of units of different types required to generate the top *N* surnames in the United States [34].

For this reason, many practical diphone systems are not purely diphone based: they do not store transitions between fricatives, or between fricatives and stops, while they store longer units that have a high level of coarticulation [48]. If only one representative of a dyad is available, there are pitch discontinuities. Prosody modification techniques, such as those described in Section 16.4, can be applied to correct this problem. Otherwise many instances of each diphone are needed for good prosodic coverage. Diphones are trainable, generalizable and offer better quality than context-independent phones.

16.3.1.3. Context-Dependent Phoneme

Another subword unit used in the literature [24] is the *context-dependent* phoneme. If the context is limited to the immediate left and right phonemes, the unit is known as *triphone*. As in speech recognition, not all N^3 need to be stored, because not all combinations will occur in practice. For English, typically there can be in excess of 25,000 triphones: 12,000 within-word triphones and another 12,000 across-word triphones. Because of the increased number of units, more contextual variations can be accommodated this way. Drawing from experience in speech recognition, we know that many different contexts have a similar effect on the phoneme; thus, several triphones can be clustered together into a smaller number of *generalized triphones*, typically between 500 and 3000. All the clustering procedures described in Chapter 9 can be used here as well. In particular, decision-tree clustered phones have been successfully used. Because a larger number of units can be used, discontinuities can be smaller than in the case of diphones while making the best use of the available data. In addition to only considering the immediate left and right phonetic context, we could also add stress for the current phoneme and its left and right context, word-dependent phones (where phones in particular words are considered distinct context-dependent phones), quinphones (where two immediate left and right phones are used), and different prosodic patterns (pitch ranges and/or durations). As in speech recognition, clustered context-dependent triphones are trainable and generalizable.

Traversing the tree for a given phoneme is equivalent to following the answers for the branching nodes from root to leaves, which determines the clusters for similar context-dependent phones. The decision trees are generated automatically from the analysis database to obtain minimum within-unit distortion (or entropy) for each split. Therefore, one must be able to acquire a large inventory of context-dependent phone HMMs with a decent coverage of the contexts one wishes to model. All the context-dependent phone units can be well replaced by any other units within the same cluster. This method generalizes to contexts not seen in the training data, because the decision tree uses questions involving broad phonetic categories of neighboring contexts, yet provides detailed models for contexts that are represented in the database. Given the assumption that these clustering decision trees should be consistent across different speakers, the use of ample speaker-independent databases instead of limited speaker-dependent databases allows us to model more contexts as well as deeper trees to generate a high-quality TTS voice. These techniques also facilitate the creation of acoustic inventories with a scalable number of units that trade off size with quality. Thus, we can use questions (about the immediate left/right phonetic contexts, stress, pitch, duration, word, etc) in the decision-tree clustering methods of Chapter 4 to reduce all the possible combinations to a manageable number.

16.3.1.4. Subphonetic Unit

Subphonetic units, or senones, have also been used with some success [13]. Typically, each phoneme can be divided into three states, which are determined by running a speech recognition system in forced alignment mode. These states can also be context dependent and can also be clustered using decision trees like the context-dependent phonemes. The HMM state has proved to be more effective than the context-dependent phone in speech recognition, also trainable and generalizable, but for synthesis it means having more concatenations and thus possibly more discontinuities. If multiple instances per subphonetic unit are used, higher quality can be obtained.

A *half phone* goes either from the middle of a phone to the boundary between phones or from the boundary between phones to the middle of the phone. This unit offers more flexibility than a phone and a diphone and has been shown useful in systems that use multiple instances of the unit [7].

16.3.1.5. Syllable

It has been observed that discontinuities across syllables stand out more than discontinuities within syllables [55], so syllables are natural units. There may be around 10,000 syllables in English, depending on the exact definition of syllable, so even a context-independent syllable system needs to store at least as many if one instance per syllable is needed for full generalizability. There will still be spectral discontinuities, though hopefully not too noticeable. More than one instance per unit may be needed to account for varying acoustic contexts or varying prosodic patterns, particularly if no waveform modification is to be used.

16.3.1.6. Word and Phrase

The unit can be as large as a word or even a phrase. While using these long units can increase naturalness significantly, generalizability and trainability are poor, so that it is difficult to have all the instances desired to synthesize an output utterance. One advantage of using a word or longer unit over its decomposition in phonemes, as in the above units, is that there is no dependence on a phonetically transcribed dictionary. It is possible that the phoneme string associated to a word by our dictionary is not correct, or not fluent enough, so that using a whole-word model can solve this problem. Of course, the system may have a combination of all units: a set of the most frequent words, sentences, or phrases for best quality some percentage of the time, and some smaller units for full generalizability and trainability.

16.3.2. Optimal Unit String: The Decoding Process

The goal of the decoding process is to choose the optimal string of units for a given phonetic string that best matches the desired prosody. Sometimes there is only one possible string, so that this process is trivial, but in general there are several strings of units that result in the same phonetic string yet some of them sound better than others. The goal is to come up with an objective function that approximates this sound quality that allows us to select the best string. The quality of a unit string is typically dominated by spectral and pitch discontinuities at unit boundaries. Discontinuities can occur because of:

1. *Differences in phonetic contexts*. A speech unit was obtained from a different phonetic context than that of the target unit.

2. Incorrect *segmentation*. Such segmentation errors can cause spectral discontinuities even if they had the same phonetic context.
3. *Acoustic* variability. Units can have the same phonetic context and be properly segmented, but variability from one repetition to the next can cause small discontinuities. A unit spoken in fast speech is generally different from another in slow or normal speech. Different recording conditions (amplitude, microphone, sound card) can also cause spectral discontinuities.
4. Different *prosody*. Pitch discontinuity across unit boundaries is also a cause for degradation.

The severity of such discontinuities generally decreases as the number of units increases. More importantly, the prosody of the concatenation has, in general, no resemblance with the prosody specified by the TTS front-end unless we have several instances of each unit, each with a different prosody, or use a prosody modification algorithm (see Section 16.4).

16.3.2.1. Objective Function

Our goal is to come up with a numeric measurement for a concatenation of speech segments that correlates well with how well they sound. To do that we define unit cost and transition cost between two units.

Let θ be a speech segment whose phonetic transcription $p = p(\theta)$. Let $\Theta = \{\theta_1, \theta_2, \cdots, \theta_N\}$ be a concatenation of N speech segments whose combined phonetic transcription is $P = \{p_1, p_2, \cdots, p_N\}$. P is a string of M phonemes, and since each segment has at least one phoneme, it holds that $M \ge N$.

For example, the phonetic string P = "*hh ax l ow*" corresponding to the word *hello* has M = 4 phonemes and can be decomposed in N = 4 segments $\Theta_1 = \{\theta_1, \theta_2, \theta_3, \theta_4\}$, where $p(\theta_1) = /hh/$, $p(\theta_2) = /ax/$, $p(\theta_3) = /l/$, $p(\theta_4) = /ow/$, each segment being a phoneme. Or it can be decomposed into N = 2 segments $\Theta_2 = \{\theta_5, \theta_6\}$, where $p(\theta_5) = /hh\ ax/$, $p(\theta_5) = /l\ ow/$, so that each segment has two phonemes. There are 8 possible such decompositions for this example (in general there are 2^{M-1} possible decompositions[1]).

The distortion or cost function between the segment concatenation Θ and the target T can be expressed as a sum of the corresponding unit costs and transition costs [27, 46] as follows:

$$d(\Theta, T) = \sum_{j=1}^{N} d_u(\theta_j, T) + \sum_{j=1}^{N-1} d_t(\theta_j, \theta_{j+1}) \tag{16.2}$$

[1] This assumes that one instance per unit is available. If there are several instances per unit, the number of decompositions grows exponentially.

where $d_u(\theta_j, T)$ is the *unit cost* of using speech segment θ_j within target T and $d_t(\theta_j, \theta_{j+1})$ is the transition cost in concatenating speech segments θ_j and θ_{j+1}. The optimal speech segment sequence of units $\hat{\Theta}$ can be found as the one that minimizes the overall cost

$$\hat{\Theta} = \arg\min_{\Theta} d(\Theta, T) \tag{16.3}$$

over sequences with all possible number of units. Transition cost and unit costs are described in Sections 16.3.2.2 and 16.3.2.3.

Let's analyze the second term in the sum of Eq. (16.2), also shown in Figure 16.6. If all transition costs were identical, the word string with fewest units would have lowest distortion. In practice transition costs are different and, thus, the string with fewest units is not necessarily the best, though there is clearly a positive correlation.

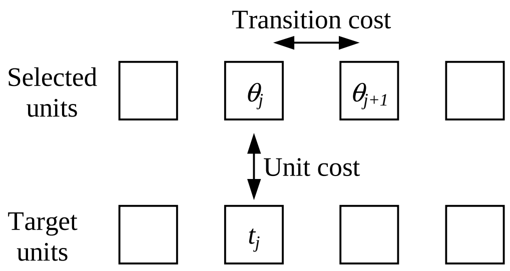

Figure 16.6 Tradeoff between unit and transition costs.

When a large number of speech segments are available, finding the segment sequence with lowest cost is a search problem like those analyzed in Chapter 12. Often a Viterbi algorithm is needed to make this efficient.

The art in this procedure is in the exact definition of both transition and unit costs, for which no standard has been defined that works best to date. In Sections 16.3.2.2 and 16.3.2.3 we present an approach for which both transition and unit costs are empirically set after perceptual experiments. Such a system is easy to build, and study of those costs gives insight into the perceptual effects.

Costs obtained using a data-driven criterion are described in Sections 16.3.2.4 and 16.3.2.5. While more complicated than that of empirical costs, this method addresses the shortcomings of the previous method. Finally, we need to estimate some weights to combine the different costs for spectrum and prosody, which can be done empirically or by regression [26].

16.3.2.2. Empirical Transition Cost

If spoken in succession, two speech segments have a zero transition cost. But, when they are excised from separate utterances, their concatenation can have varying degrees of naturalness. The transition cost incorporates two types of continuity measures: coarticulatory and prosodic.

An approximation to the prosodic continuity measure is to make it proportional to the absolute difference of the F0 or log F0 at the boundaries, if the boundary is voiced for both units. If we use the prosody modification techniques of Section 16.4, this cost could be set to a small value to reflect the fact that prosody modification is not a perfect process. More sophisticated cost functions can be used to account for prosody mismatches [10].

Regarding the coarticulatory effect, it has been empirically observed that a concatenation within a syllable is more perceptible than when the concatenation is at the syllable boundary. Yi [55] proposed an empirical cost matrix for the concatenation of two speech segments when that concatenation occurs within a syllable (Table 16.5) or at a syllable boundary (Table 16.6). Phonemes are grouped by manner of articulation: vowel/semivowels, fricatives, stops, and nasals. The rows represent the left side of the transition and the columns represent the right side, and NA represents a case that does not occur. These costs reflect perceptual ratings by human listeners to unit concatenations between different phonemes. Values of 10, 2000, 5000, 7500, and 10,000 were used to indicate different degrees of goodness from very good to very bad concatenations.

Table 16.5 Cost matrix for intrasyllable concatenations (after Yi [55]). The rows represent the left side of the transition and the columns represent the right side, and NA represents a case that does not occur.

	vowel	semivowel	nasal	obstruent	/h/
vowel	10,000	10,000	7500	10	NA
semivowel	10,000	7500	7500	10	NA
nasal	5000	10	NA	10	NA
/h/	5000	NA	NA	NA	NA
obstruent	10	10	10	10,000	NA

Table 16.6 Cost matrix for intersyllable concatenations (after Yi [55]). The rows represent the left side of the transition and the columns represent the right side, and NA represents a case that does not occur.

	vowel	semivowel	nasal	obstruent	/h/	Silence
vowel	NA	7500	5000	10	5000	10
semivowel	7500	7500	2000	10	10	10
nasal	2000	10	10	10	10	10
obstruent	10	10	10	5000	10	10
/h/	NA	NA	NA	NA	NA	NA
silence	10	10	10	10	10	10

16.3.2.3. Empirical Unit Cost

The unit cost is generally a combination of the coarticulation cost and the prosodic cost. Prosodic mismatches can be made proportional to the F0 difference between the candidate unit and the target unit or set to a fixed low value if the prosody modification techniques of Section 16.4 are used.

A way of determining the cost associated with replacing a phonetic context with another was proposed by Yi [55], who empirically set cost matrices for phone classes by listening to concatenations where such contexts were replaced. These ad hoc values also bring some sense of where the coarticulation problems are. Replacing a vowel or semivowel by another with a context that has a different place of articulation or nasalization results in audible discontinuities. The rows represent the context class of the target phoneme and the columns represent the context class of the proposed unit. Table 16.7, Table 16.8, Table 16.9, Table 16.10, Table 16.11, and Table 16.12 include an empirical set of costs for such mismatches between the target's context and a candidate unit's context for the case of vowel/semivowels, fricatives, stops, and nasals. These costs reflect human listeners' perceptual ratings of speech units with an incorrect phonetic context. Values of 10, 100, 500, and 1000 were used to indicate very good, good, bad, and very bad units. These values are chosen to match the values for transition costs of Section 16.3.2.2.

Table 16.7 Unit coarticulation cost matrix (after Yi [55]) for left and right context replacements for vowels and semivowels.

	labial	alv/den/pal	velar	m	n	ng	front	back	none
labial	10	1000	1000	1000	1000	1000	1000	1000	1000
alv/den/pal	1000	10	1000	1000	1000	1000	1000	1000	1000
velar	1000	1000	10	1000	1000	1000	1000	1000	1000
m	1000	1000	1000	10	1000	1000	1000	1000	1000
n	1000	1000	1000	1000	10	1000	1000	1000	1000
ng	1000	1000	1000	1000	1000	10	1000	1000	1000
front	1000	1000	1000	1000	1000	1000	10	1000	1000
back	1000	1000	1000	1000	1000	1000	1000	10	1000
none	1000	1000	1000	1000	1000	1000	1000	1000	10

Table 16.8 Unit coarticulation cost matrix (after Yi [55]) for left and right context replacements for fricatives.

	retroflex	round	sonorant	other
retroflex	10	100	100	100
round	100	10	100	100
sonorant	100	100	10	100
other	100	100	100	10

Table 16.9 Unit coarticulation cost matrix (after Yi [55]) for left context replacements for stops.

	front	back	retroflex	round	other
front	10	10	10	10	10
back	10	10	10	10	10
retroflex	10	10	10	10	10
round	10	10	10	10	10
other	500	500	500	500	10

Table 16.10 Unit coarticulation cost matrix (after Yi [55]) for right context replacements for stops.

	front	back	retroflex	round	schwa	other
Front	10	100	100	100	500	100
Back	100	10	100	100	500	100
Retroflex	100	100	10	100	500	100
Round	100	100	100	10	500	100
Schwa	500	500	500	500	10	500
Other	100	100	100	100	500	10

Table 16.11 Unit coarticulation cost matrix (after Yi [55]) for left context replacements for nasals.

	obstruent	sonorant
obstruent	10	1000
sonorant	1000	10

Table 16.12 Unit coarticulation cost matrix (after Yi [55]) for right context replacements for nasals.

	voiced	unvoiced	sonorant
voiced	10	100	1000
unvoiced	100	10	1000
sonorant	1000	1000	10

16.3.2.4. Data-Driven Transition Cost

The empirical transition costs of Section 16.3.2.2 do not necessarily mean that a spectral discontinuity will take place, only that one is likely, and that if it occurs within a syllable it will have a larger perceptual effect than if it occurs across syllable boundaries. While that method can result in a good system, the cost is done independently of whether there is a true spectral discontinuity or not. Thus, it has been also proposed to use a measurement of the spectral discontinuity directly. This is often estimated as:

$$d_t(\theta_i, \theta_j) = \left| \mathbf{x}_i(l(\theta_i) - 1) - \mathbf{x}_j(0) \right|^2 \tag{16.4}$$

the magnitude squared of the difference between the cepstrum at the last frame of θ_i and the first frame of θ_j. The quantity $l(\theta_i)$ denotes the number of frames of speech segment θ_i, and $\mathbf{x}_i(k)$ the cepstrum of segment θ_i at frame *k*.

This technique can effectively measure a spectral discontinuity in a region with slowly varying spectrum, but it can fail when one of the segments is a nasal, for example, for which a sharp spectral transition is expected and desired. A better way of measuring this discontinuity is shown in Figure 16.7, in which we measure the cepstral distance in an overlap region:[2] the last frame of segment 1 and the first frame before the beginning of segment 2:

$$d_t(\theta_i, \theta_j) = \left| \mathbf{x}_i(l(\theta_i) - 1) - \mathbf{x}_j(-1) \right|^2 \tag{16.5}$$

When many speech segments are considered, a large number of cepstral distances need to be computed, which in turn may result in a slow process. To speed it up an approximation can be made where all possible cepstral vectors at the boundaries are vector quantized first, so that the distances between all codebook entries can be precomputed and stored in a table.

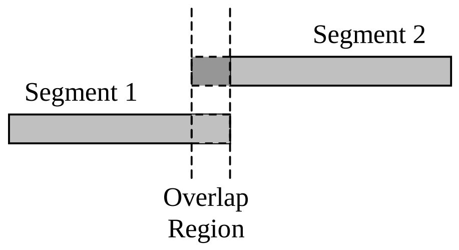

Figure 16.7 Measurement of the spectral discontinuity in the overlap region. The dark gray area is the speech region that precedes segment 2 and does not form part of segment 2. This area should match the last part of the segment 1.

A spectral discontinuity across, say, fricatives is perceptually not as important as if they happen across vowels [48]. For this reason, the cepstral distance described above does

[2] This means extra frames need to be stored.

not correlate well with perceptual distances. To solve this problem, it is possible to combine both methods, for example by weighting the spectral/cepstral distance by different values.

Even if no spectral discontinuity is present, a phase discontinuity may take place. The pitch periodicity may be lost at the boundary. This can be generally solved by fine adjustment of the boundary using a correlation approach as described in Section 16.4. You need to keep in mind that such methods are not perfect.

16.3.2.5. Data-Driven Unit Cost

Spectral discontinuities across concatenations are often the result of using a speech segment with a different phonetic context than the target. One possibility is, then, to consider only speech segments where the phonetic contexts to the left and right match exactly. For example, if we use a phoneme as the basic speech segment, a perfect match would require on the order of at least 25000 different segments. In this case, the coarticulation unit cost is zero if the target and candidate segment have the same phonetic context and infinite otherwise. When longer segments are desired, this number explodes exponentially. The problem with this approach is that it severely reduces the number of potential speech segments that can be used.

Generalized triphones, as described in Section 16.3.1.3, are used in [24]. In this approach, if the speech segments have the same generalized triphone contexts as the target utterance, the unit cost is zero, otherwise the cost is infinite. The technique allows us to use many more possible speech segments than the case above, yet it eliminates those speech segments that presumably have context mismatches that in turn lead to unnatural concatenations. When using a large training database, it was found that bringing the number of triphones from 25,000 down to about 2000 did not adversely impact the quality, whereas some degradation was perceived when using only 500 phoneme-length segments. Thus, this technique allows us to reduce the size of the speech segment inventory without severely degrading the voice quality.

If we set the number of decision-tree clustered context-dependent phonemes to be large, there will be fewer choices of long speech segments that match. For instance, in a system with 2000 generalized triphones, the phonetic context of the last phoneme of a long segment and the context of the target phoneme may be clustered together, whereas in a 3000-generalized-triphone system, both contexts may not be clustered together, so that the long segment cannot be used. This would be one example where using a larger number of generalized triphones hurts speech naturalness because the database of speech segments is limited. This problem could have been avoided if we didn't have to match generalized triphones and instead allowed context substitutions, yet penalized them with a corresponding cost. In the framework of decision-tree clustered context-dependent phonemes, this cost can be computed as the increase in entropy when those contexts are merged, using the methods described in Chapter 9. The larger the increase in entropy, the larger the penalty is when doing that context substitution between the candidate segment and the target segment. This approach gives more flexibility in the number of speech segments to be considered. In this case, there is a nonzero unit coarticulation cost associated with replacing one phonetic context with another.

Speech segments that have low HMM probability can be discarded, as they are probably not representative enough for that unit. Moreover, we can eliminate outliers: those units that have parameters too far away from the mean. Eliminating pitch outliers helps if prosody modification is to be done, as modifying pitch by more than a factor of 2 typically yields a decrease of quality, and by keeping units with *average* pitch, this is less likely to occur. Eliminating duration or amplitude outliers may signal an incorrect segmentation or a bad transcription [13].

16.3.3. Unit Inventory Design

The minimal procedure to obtain an acoustic inventory for a concatenative speech synthesizer consists of simply recording a number of utterances from a single speaker and labeling them with the corresponding text.

Since recording is often done in several sessions, it is important to maintain the recording conditions constant to avoid spectral or amplitude discontinuities caused by changes in recording conditions. The same microphone, room, and sound card should be used throughout all sessions [49].

Not all donor speakers are created equal. The choice of donor speaker can have a significant effect in voice quality (up to 0.3 MOS points on a 5-MOS scale) [7, 51, 52].

We can obtain higher-quality concatenative synthesis if the text read by the target speaker is representative of the text to appear in our application. This way we will be able to use longer units, and few concatenations will be needed.

Then the waveforms have to be segmented into phonemes, which is generally done with a speech recognition system operating in forced-alignment mode. Phonetic transcription, including alternate pronunciations, is generated automatically from text by the phonetic analysis module of Chapter 14. A large part of the inventory preparation includes checking correspondence between the text and corresponding waveform. Possible transcription errors may be flagged by phonemes whose durations are too far away from the mean (outliers) [13, 24].

Once we have the segmented and labeled recordings, we can use them as our inventory, or create smaller inventories as subsets that trade off memory size with quality [21, 25]. A database with a large number of utterances is generally required to obtain high-quality synthesis. It is noteworthy to analyze whether we can reduce the size of our database while obtaining similar synthesis quality on a given set of utterances. To do this, we can measure the cost incurred when we use a subset of the units in the database to synthesize our training database. A greedy algorithm can be used that at each stage eliminates the speech unit that increases the total distortion the least, repeating the approach until the desired size is achieved. This is an iterative analysis-by-synthesis algorithm.

The above procedure can also be used to find the set of units that have lowest cost in synthesizing a given text. For efficiency, instead of a large training text, we could use representative information from such text corpus, like the word trigrams with their corresponding counts, as an approximation.

In concatenative systems, you need to store a large number of speech segments, which could be compressed using any of the speech coding techniques described in Chapter 7.

Since many such coders encode a frame of speech based on the previous one, you need to store this context for every segment you want to encode if you are to use such systems.

16.4. PROSODIC MODIFICATION OF SPEECH

One problem of segment concatenation is that it doesn't generalize well to contexts not included in the training process, partly because prosodic variability is very large. There are techniques that allow us to modify the prosody of a unit to match the target prosody. These prosody-modification techniques degrade the quality of the synthetic speech, though the benefits are often greater than the distortion introduced by using them because of the added flexibility.

The objective of prosodic modification is to change the amplitude, duration, and pitch of a speech segment. Amplitude modification can be easily accomplished by direct multiplication, but duration and pitch changes are not so straightforward.

We first present OLA and SOLA, two algorithms to change the duration of a speech segment. Then we introduce PSOLA, a variant of the above that allows for pitch modification as well.

16.4.1. Synchronous Overlap and Add (SOLA)

Time-scale modification of speech is very useful, particularly voice compression, as it allows a user to listen to a voice mail or taped lecture in a fraction of the time taken by the original segment user to listen to information The overlap-and-add (OLA) technique [12] shown in Figure 16.8 shows the analysis and synthesis windows used in the time compression. Given a Hanning window of length $2N$ and a compression factor of f, the analysis windows are spaced fN. Each analysis window multiplies the analysis signal, and at synthesis time they are overlapped and added together. The synthesis windows are spaced N samples apart. The use of windows such as Hanning allows perfect reconstruction when f equals 1.

In Figure 16.8, some of the signal appearance has been lost; note particularly some irregular pitch periods. To solve this problem, the synchronous overlap-and-add (SOLA) [45] allows for a flexible positioning of the analysis window by searching the location of the analysis window i around fNi in such a way that the overlap region had maximum correlation. The SOLA algorithm produces high-quality time compression. A mathematical formulation of PSOLA, an extension of both OLA and SOLA, is presented in Section 16.4.2.

While typically compression algorithms operate at a uniform rate, they have also been used in a nonuniform rate to take into account human perception, so that rapid transitions are compressed only slightly, steady sounds are compressed more, and pauses are compressed the most. It's reported in [11], that while uniform time compression can achieve a factor of 2.5 at most without degradation in intelligibility, a nonuniform compression allows up to an average compression factor of 4.

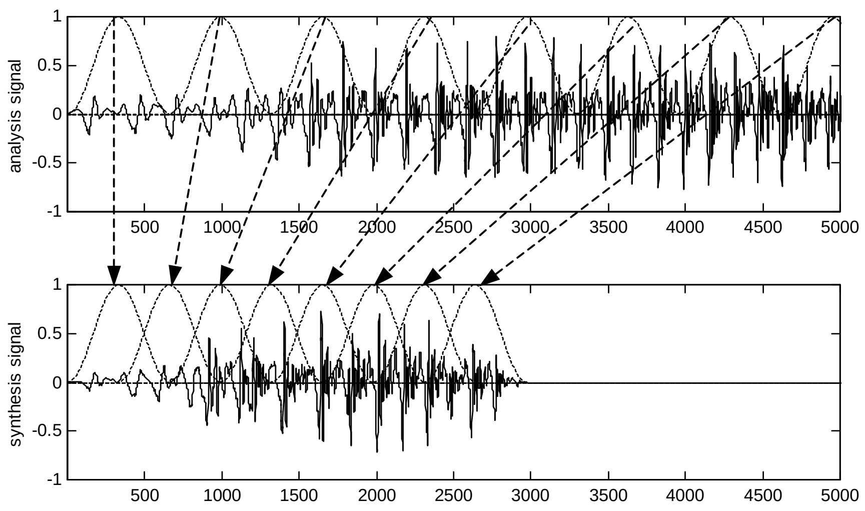

Figure 16.8 Overlap-and-add (OLA) method for time compression. Hanning windows, N = 330, are used to multiply the analysis signal, and resulting windowed signals are added. The analysis windows, spaced $2N$ samples, and the analysis signal $x[n]$ are shown on the top. The synthesis windows, spaced N samples apart, and the synthesis signal $y[n]$ are shown below. Time compression is uniform with a factor of 2. Pitch periodicity is somewhat lost, particularly around the fourth window.

16.4.2. Pitch Synchronous Overlap and Add (PSOLA)

Both OLA and SOLA do duration modification but cannot do pitch modification. On the other hand, they operate without knowledge of the signal's pitch. The most widely used method to do pitch modification is called pitch synchronous overlap and add (PSOLA) [38, 39], though to do so it requires knowledge of the signal's pitch. This process is illustrated in Figure 16.9.

Let's assume that our input signal $x[n]$ is voiced, so that it can be expressed as a function of pitch cycles $x_i[n]$

$$x[n] = \sum_{i=-\infty}^{\infty} x_i[n - t_a[i]] \tag{16.6}$$

where $t_a[i]$ are the epochs of the signal, so that the difference between adjacent epochs $P_a[i] = t_a[i] - t_a[i-1]$ is the pitch period at time $t_a[i]$ in samples. The pitch cycle is a windowed version of the input

$$x_i[n] = w_i[n]x[n] \tag{16.7}$$

which requires the windows $w_i[n]$ to meet the following condition:

$$\sum_{i=-\infty}^{\infty} w_i[n - t_a[i]] = 1 \tag{16.8}$$

which can be accomplished with a Hanning window, or a trapezoidal window that spans two pitch periods.

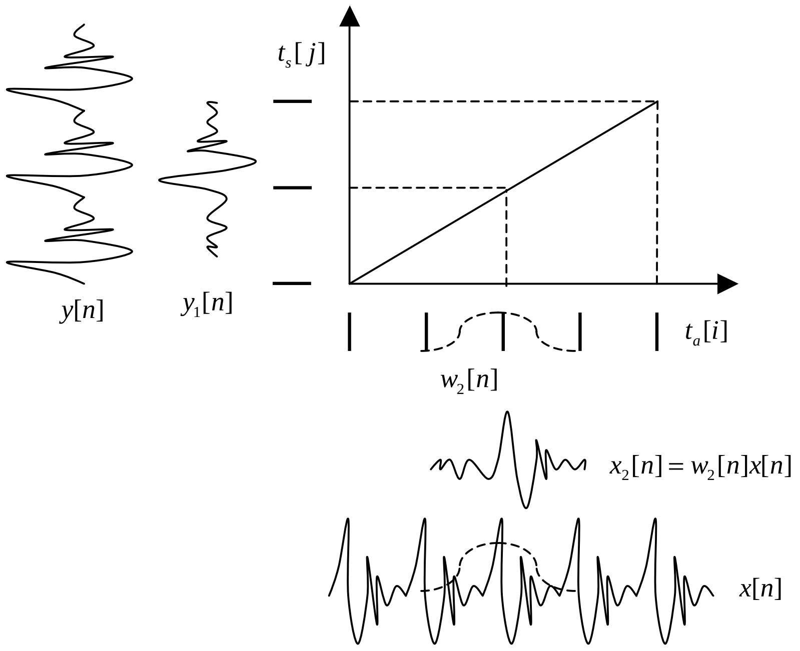

Figure 16.9 Mapping between five analysis epochs $t_a[i]$ and three synthesis epochs $t_s[j]$. Duration has been shortened by 40% and pitch period increased by 60%. Pitch cycle $x_2[n]$ is the product of the analysis window $w_2[n]$, in dotted line, with the analysis signal $x[n]$, which is aligned with analysis epochs $t_a[i]$. In this case, synthesis pitch cycle $y_1[n]$ equals $x_2[n]$ and also $y_0[n] = x_0[n]$ and $y_2[n] = x_5[n]$. Pitch is constant over time in this case.

Our goal is to synthesize a signal $y[n]$, which has the same spectral characteristics as $x[n]$ but with a different pitch and/or duration. To do this, we replace the analysis epoch sequence $t_a[i]$ with the synthesis epochs $t_s[j]$, and the analysis pitch cycles $x_i[n]$ with the synthesis pitch cycles $y_j[n]$:

$$y[n] = \sum_{j=-\infty}^{\infty} y_j[n - t_s[j]] \tag{16.9}$$

The synthesis epochs are computed so as to meet a specified duration and pitch contour, as shown in Figure 16.9. This is equivalent to an impulse train with variable spacing driving a time-varying filter $x_t[n]$ which is known for $t = t_a[i]$, as shown in Figure 16.10. The synthesis pitch cycle $y_j[n]$ is obtained via a mapping from the closest corresponding analysis pitch cycle $x_i[n]$. In the following sections we detail how to calculate the synthesis epochs and the synthesis pitch-cycle waveforms.

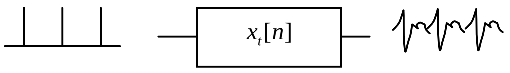

Figure 16.10 PSOLA technique as an impulse train driving a time-varying filter.

The term overlap-and-add derives from the fact that we use overlapping windows that we add together. The pitch-synchronous aspect comes from the fact that the windows are spaced a pitch period apart and are two pitch periods long. As you can see from Figure 16.9, the synthesis waveform has a larger pitch period than the analysis waveform and is shorter in duration.

For unvoiced speech, a set of epochs that are uniformly spaced works well in practice, as long as the spacing is smaller than 10 ms. If the segment needs to be stretched in such a way that these characteristic waveforms are repeated, an artificial periodicity would appear. To avoid this, the characteristic waveform that was to be repeated is flipped in time [38].

This approach is remarkably simple, yet it leads to high-quality prosody modification, as long as the voiced/unvoiced decision is correct and the epoch sequence is accurate.

To do prosody modification, the PSOLA approach requires keeping the waveform of the speech segment and its corresponding set of epochs, or time marks. As you can see from Eq. (16.6), if no prosody modification is done, the original signal is recovered exactly.

16.4.3. Spectral Behavior of PSOLA

Let's analyze why this simple technique works and how. To do that let's consider the case of a speech signal $x[n]$ that is exactly periodic with period T_0 and can be created by passing an impulse train through a filter with impulse response $s[n]$:

$$x[n] = s[n] * \sum_{i=-\infty}^{\infty} \delta[n - iT_0] = \sum_{i=-\infty}^{\infty} s[n - iT_0] \tag{16.10}$$

If we know the impulse response $s[n]$, then we could change the pitch by changing T_0. The problem is how to estimate it from $x[n]$. Let's assume we want to build an estimate $\tilde{s}[n]$ by multiplying $x[n]$by a window $w[n]$:

$$\tilde{s}[n] = w[n]x[n] \tag{16.11}$$

The Fourier transform of $x[n]$ in Eq. (16.10) is given by

$$X(\omega) = S(\omega) \sum_{k=-\infty}^{\infty} \delta(\omega - k\omega_0) = \sum_{k=-\infty}^{\infty} S(\omega_0)\delta(\omega - k\omega_0) \tag{16.12}$$

where $\omega_0 = 2\pi / T_0$. The Fourier transform of $\tilde{s}[n]$ can be obtained using Eqs. (16.11) and (16.12):

$$\tilde{S}(\omega) = W(\omega) * X(\omega) = \sum_{k=-\infty}^{\infty} S(\omega_0) W(\omega - k\omega_0) \tag{16.13}$$

If the window $w[n]$ is pitch synchronous, a rectangular window with length T_0 or a Hanning window with length $2T_0$, for example, then the above estimate is exact at the harmonics, i.e., $\tilde{S}(k\omega_0) = S(k\omega_0)$, because the window leakage terms are zero at the harmonics. In-between harmonics, $\tilde{S}(\omega)$ is an interpolation using $W(\omega)$, the transfer function of the window. If we use a rectangular window, the values of $S(\omega)$ in between $S(k\omega_0)$ and $S((k+1)\omega_0)$ are not determined only by those two harmonics, because the leakage from the other harmonics are not negligible. The use of a Hanning window drastically attenuates this leakage, so the estimate of the spectral envelope is better. This is what PSOLA is doing: getting an estimate of the spectral envelope by using a pitch-synchronous window.

Since it is mathematically impossible to recover $S(\omega)$ for a periodic signal, it is reasonable to fill in the remaining values by interpolation with the main lobes of the transform of the window. This approach works particularly well if the harmonics form a dense sampling of the spectral envelope, which is the case for male speakers. For female speakers, where the harmonics may be spaced far apart, the estimated spectral envelope could be far different from the real envelope.

16.4.4. Synthesis Epoch Calculation

In practice, we want to generate a set of synthesis epochs $t_s[j]$ given a target pitch period $P_s(t)$. If the desired pitch period $P_s(t) = P$ is constant, then the synthesis epochs are given by $t_s[j] = jP$.

In general the desired pitch period $P_s(t)$ is a function of time. Intuitively, we could compute $t_s[j+1]$ in terms of the previous epoch $t_s[j]$ and the pitch period at that time:

$$t_s[j+1] - t_s[j] = P_s(t_s[j]) \tag{16.14}$$

though this is an approximation, which happens to work well if $P_s(t)$ changes slowly over time.

Now we derive an exact equation, which also can help us understand pitch-scale and time-scale modifications of the next few sections. Epoch $t_s[j+1]$ can be computed so that the distance between adjacent epochs $t_s[j+1] - t_s[j]$ equals the average pitch period in the region $t_s[j] \le t < t_s[j+1]$ between them (see Figure 16.11). This can be done by the following expression

$$t_s[j+1]-t_s[j]=\frac{1}{t_s[j+1]-t_s[j]}\int_{t_s[j]}^{t_s[j+1]}P_s(t)dt \tag{16.15}$$

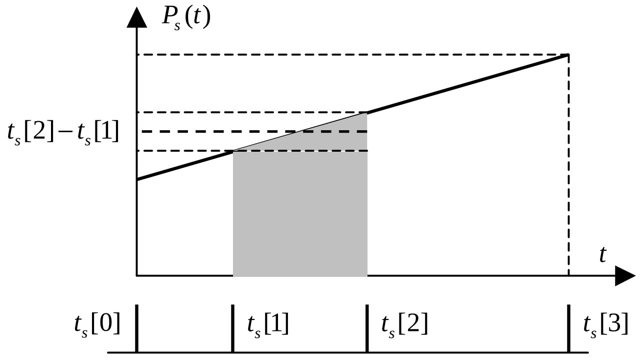

Figure 16.11 The desired pitch period $P_s(t)$ is a linearly increasing function of time such that the pitch period is doubled by the end of the segment. The four synthesis epochs $t_s[j]$ are computed to satisfy Eq. (16.15). In particular, $t_s[2]$ is computed such that $t_s[2]-t_s[1]$ equals the average pitch period in that region. Note that the growing spacing between epochs indicates that pitch is growing over time.

It is useful to consider the case of $P_s(t)$ being linear with *t* in that interval:

$$P_s(t)=P_s\left(t_s[j]\right)+b\left(t-t_s[j]\right) \tag{16.16}$$

so that the integral in Eq. (16.15) is given by

$$\int_{t_s[j]}^{t_s[j+1]}P_s(t)dt=\delta_j\left[P(t_s[i])+b\frac{\delta_j}{2}\right] \tag{16.17}$$

where we have defined δ_j as

$$\delta_j=t_s[j+1]-t_s[j] \tag{16.18}$$

Inserting Eqs. (16.17) and (16.18) into Eq. (16.15), we obtain

$$\delta_j=P(t_s[i])+b\frac{\delta_j}{2} \tag{16.19}$$

which, using Eq. (16.18), gives a solution for epoch $t_s[j+1]$ as

$$t_s[j+1]-t_s[j]=\delta_j=\frac{P_s(t_s[j])}{\left(1-b/2\right)} \tag{16.20}$$

from the previous epoch $t_s[j]$, the target pitch at that epoch $P_s(t_s[j])$, and the slope *b*. We see that Eq. (16.14) is a good approximation to Eq. (16.20) if the slope *b* is small.

Evaluating Eq. (16.16) for $t_s[j+1]$ results in an expression for $P_s(t_s[j+1])$

$$P_s(t_s[j+1]) = P_s\left(t_s[j]\right) + b\left(t_s[j+1] - t_s[j]\right) \tag{16.21}$$

Equations (16.20) and (16.21) can be used iteratively. It is important to note that Eq. (16.20) requires $b < 2$ in order to obtain meaningful results, which fortunately is always the case in practice.

When synthesizing excitations for speech synthesis, it is convenient to specify the synthesis pitch period $P_s(t)$ as a piecewise linear function of time. In this case, Eq. (16.20) is still valid as long as $t_s[j+1]$ falls within the same linear segment. Otherwise, the integral in Eq. (16.17) has two components, and a second-order equation needs to be solved to obtain $t_s[j+1]$.

16.4.5. Pitch-Scale Modification Epoch Calculation

Sometimes, instead of generating an epoch sequence given by a function $P_s(t)$, we want to modify the epoch sequence of an analysis signal with epochs $t_a[i]$ by changing its pitch while maintaining its duration intact. This is called *pitch-scale* modification. To obtain the corresponding synthesis epochs, let's assume that the pitch period $P_a(t)$ of the analysis waveform at time *t* is constant and equals the difference between both epochs

$$P_a(t) = t_a[i+1] - t_a[i] \tag{16.22}$$

as seen in Figure 16.12.

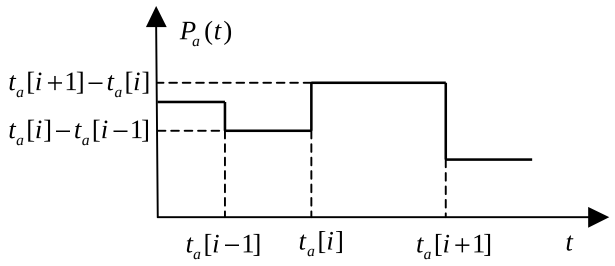

Figure 16.12 Pitch period of the analysis waveform as a function of time. It is a piece-wise constant function of time.

The pitch period of the synthesis waveform $P_s(t)$ at the same time *t* now falls in between epochs *j* and *j* + 1

$$t_s[j] \le t < t_s[j+1] \tag{16.23}$$

with $t_s[j]$ being the time instant of the *j* epoch of the synthesis waveform. Now, let's define a relationship between analysis and synthesis pitch periods

$$P_s(t) = \beta(t) P_a(t) \tag{16.24}$$

where $\beta(t)$ reflects the pitch-scale modification factor, which, in general, is a function of time. Following the derivation in Section 16.4.3, we compute the synthesis epoch $t_s[j+1]$ so that

$$t_s[j+1] - t_s[j] = \frac{1}{t_s[j+1] - t_s[j]} \int_{t_s[j]}^{t_s[j+1]} \beta(t) P_a(t) dt \tag{16.25}$$

which reflects the fact that the synthesis pitch period at time *t* is the average pitch period of the analysis waveform times the pitch-scale modification factor. Since $\beta(t)P_a(t)$ is piecewise linear, we can use the results of Section 16.4.3 to solve for $t_s[j+1]$. In general, it needs to be solved recursively, which results in a second-order equation if $\beta(t)$ is a constant or a linear function of *t*.

16.4.6. Time-Scale Modification Epoch Calculation

Time-scale modification of speech involves changing the duration of a speech segment while maintaining its pitch intact. This can be realized by defining a mapping $t_s = D(t_a)$, a time-warping function, between the original signal and the modified signal. It is useful to define the duration modification rate $\alpha(t)$ from which the mapping function can be derived:

$$D(t) = \int_0^t \alpha(\tau) d\tau \tag{16.26}$$

Let's now assume that the duration modification rate $\alpha(t) = \alpha$ is constant, so that the mapping *D*(*t*) in Eq. (16.26) is linear. If $\alpha > 1$, we are slowing down the speech, whereas if $\alpha < 1$, we are speeding it up. Let's consider time *t* in between epochs *i* and *I* + 1 so that $t_a[i] \le t < t_a[i+1]$:

$$\begin{aligned} D(t_a[0]) &= 0 \\ D(t) &= D(t_a[i]) + \alpha(t - t_a[i]) \end{aligned} \tag{16.27}$$

So that the relationship between analysis and synthesis pitch periods is given by

$$P_s(D(t)) = P_a(t) \tag{16.28}$$

To solve this it is useful to define a stream of virtual time instants $t'_a[j]$ in the analysis signal related to the synthesis time instants by

$$t_s[j] = D(t'_a[j]) \tag{16.29}$$

as shown in Figure 16.13.

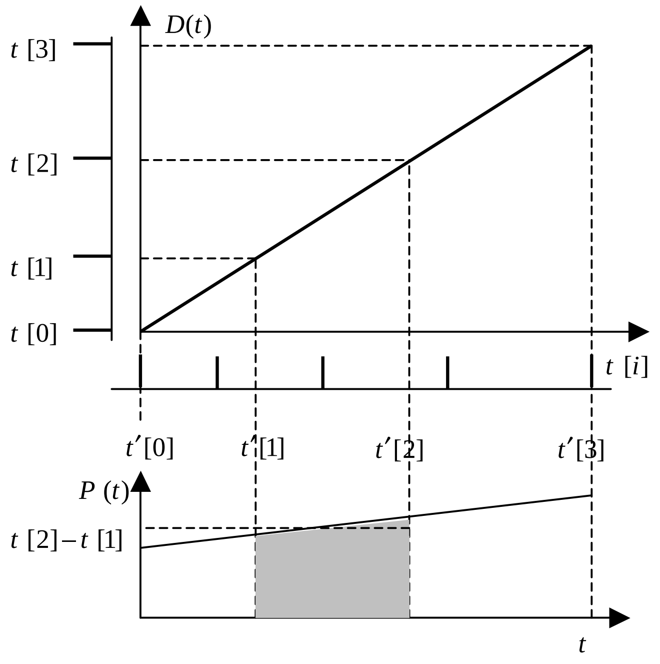

Figure 16.13 Time-scale modification of speech. The five analysis epochs $t_a[i]$ are shown in the *x*-axis and the four synthesis epochs $t_s[i]$ in the ordinate. Duration is shortened by 25% while maintaining the same pitch period. The corresponding virtual analysis epochs $t'_a[i]$ are obtained through the mapping *D*(*t*), a linear transformation with $\alpha = 0.75$.

Now we try to determine $t_s[j+1]$ such that $t_s[j+1]-t_s[j]$ is equal to the average pitch period in the original time signal between $t'_a[j]$ and $t'_a[j+1]$:

$$t_s[j+1]-t_s[j]=\frac{1}{t'_a[j+1]-t'_a[j]}\int_{t'_a[j]}^{t'_a[j+1]} P_a(t)dt \tag{16.30}$$

which, using Eqs. (16.27) and (16.29), results in

$$t_s[j+1]-t_s[j]=\frac{\alpha}{t_s[j+1]-t_s[j]}\int_{t_s[j]/\alpha}^{t_s[j+1]/\alpha} P_a(t)dt \tag{16.31}$$

which again results in a second-order equation if $P_a(t)$ is piecewise constant or linear in *t*.

16.4.7. Pitch-Scale Time-Scale Epoch Calculation

The case of both pitch-scale and time-scale modification results in a combination of Eqs. (16.25) and (16.31):

$$t_s[j+1]-t_s[j]=\frac{\alpha}{t_s[j+1]-t_s[j]}\int_{t_s[j]/\alpha}^{t_s[j+1]/\alpha}\beta(t)P_a(t)dt \tag{16.32}$$

which again results in a second-order equation if $\beta(t)P_a(t)$ is piecewise constant or linear in *t*.

16.4.8. Waveform Mapping

The synthesis pitch waveforms can be computed through linear interpolation. Suppose that $t_a[i]\le t'_a[j]<t_a[i+1]$, then $y_j[n]$ is given by

$$y_j[n]=(1-\gamma_j)x_i[n]+\gamma_j x_{i+1}[n] \tag{16.33}$$

where γ_j is given by

$$\gamma_j=\frac{t'_a[j]-t_a[i]}{t_a[i+1]-t_a[i]} \tag{16.34}$$

Using this interpolation for voiced sounds results in smooth speech. For unvoiced speech, this interpolation results in a decrease of the amount of aspiration. Since smoothness is not a problem in those cases, the interpolation formula above is not used for unvoiced frames. A simplification of this linear interpolation consists of rounding γ_j to 0 or 1 and, thus, selecting the closest frame.

16.4.9. Epoch Detection

In the PSOLA approach, the analysis epochs $t_a[i]$ were assumed known. In practice this is not the case and we need to estimate them from the speech signal. There can be errors if the pitch period is not correctly estimated, which results in a rough, noisy voice quality. But estimating the epochs is not a trivial task, and this is the most sensitive part of achieving prosody modification in PSOLA.

Most pitch trackers attempt to determine F0 and not the epochs. From the $t_a[i]$ sequence it is easy to determine the pitch, since $P(t)=t_a[i+1]-t_a[i]$ for $t_a[i]<t<t_a[i+1]$. But from the pitch *P*(*t*) the epoch placement is not uniquely determined, since the time origin is unspecified.

Common pitch tracking errors, such as pitch doubling, subharmonic selection, or errors in voiced/unvoiced decisions, result in rough speech. While manual pitch marking can result in accurate pitch marks, it is time consuming and error prone as well, so automatic methods have receivedt a great deal of attention.

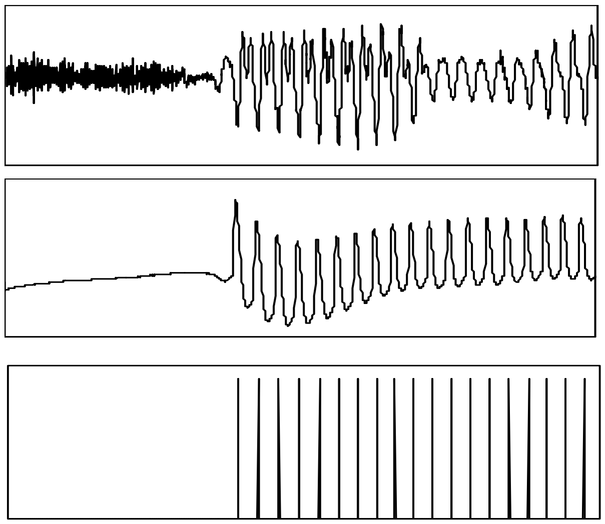

Figure 16.14 Speech signal, laryngograph signal, and its corresponding epochs.

A method that attains very high accuracy has been proposed through the use of an *e-lectroglottograph* (EGG) [32]. It consists of a pair of electrodes strapped around the neck at both sides of the larynx that measures the impedance of the larynx. Such a device, also called *laryngograph*, delivers a periodic signal when the vocal cords are vibrating and no signal otherwise. The period of a laryngograph signal is fairly stationary, which makes it relatively easy to determine the epochs from it (see Figure 16.14).

High-quality epoch extraction can be achieved by performing peak picking on the derivative of the laryngograph signal. Often, the derivative operation is accomplished by a first-order preemphasis filter $H[z]=1-\alpha z^{-1}$, with α being close to 1 (0.95 is a good choice).

In practice, the signal is preprocessed to filter out the low frequencies (lower than 100 Hz) and high frequencies (higher than 4 kHz). This can be done with rectangular window filters that are quite efficient and easy to implement. There is a significant amount of energy outside this band that does not contribute to epoch detection, yet it can complicate the process, as can be seen in Figure 16.14, so this bandpass filtering is quite important.

The preemphasized signal exhibits peaks that are found by thresholding. The quality of this epoch detector has been evaluated on recordings from two female and four male speakers, and the voiced/unvoiced decision errors are lower than 1%. This is definitely acceptable for our prosody-modification algorithms. The quality of prosody modification with the epochs computed by this method vastly exceeded the quality achieved when standard pitch trackers (as described in Chapter 6) were used on the original speech signal [2].

16.4.10. Problems with PSOLA

The PSOLA approach is very effective in changing the pitch and duration of a speech segment if the epochs are determined accurately. Even assuming there are no pitch tracking errors, there can be problems when concatenating different segments:

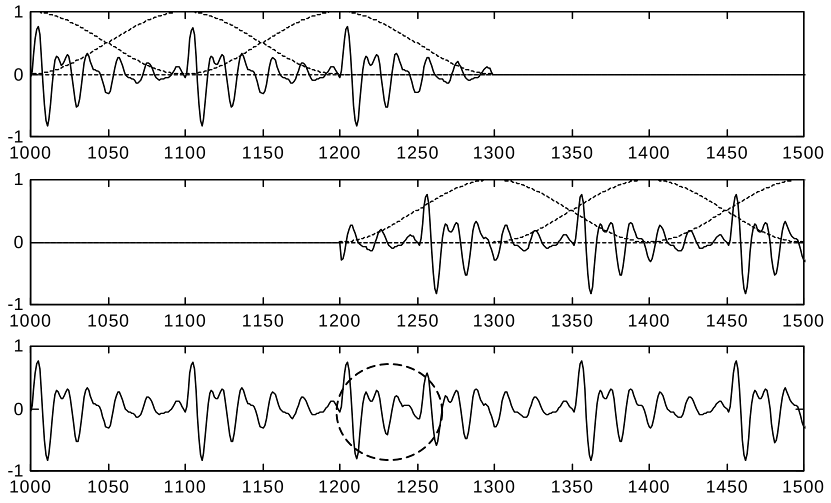

Figure 16.15 Phase mismatches in unit concatenation. Waveforms are identical, but windows are not centered on the same relative positions within periods.

Phase mismatches. Even if the pitch period is accurately estimated, mismatches in the positioning of the epochs in the analysis signal can cause glitches in the output, as can be seen in Figure 16.15.

The MBROLA [15] technique, an attempt to overcome phase mismatches, uses the time-domain PSOLA method for prosody modification, but the pitch cycles have been preprocessed so that they have a fixed phase. The advantage is that the spectral smoothing can be done by directly interpolating the pitch cycles in the time domain without adding any extra complexity. Since MBROLA sets the phase to a constant, the algorithm is more robust to phase errors in the epoch detection. Unfortunately, setting the phases constant incurs the added perceived noise described before.

Pitch mismatches. These occur even if there are no pitch or phase errors during the analysis phase. As shown in Section 16.4.3, if two speech segments have the same spectral envelope but different pitch, the estimated spectral envelopes are not the same, and, thus, a discontinuity occurs (see Figure 16.16).

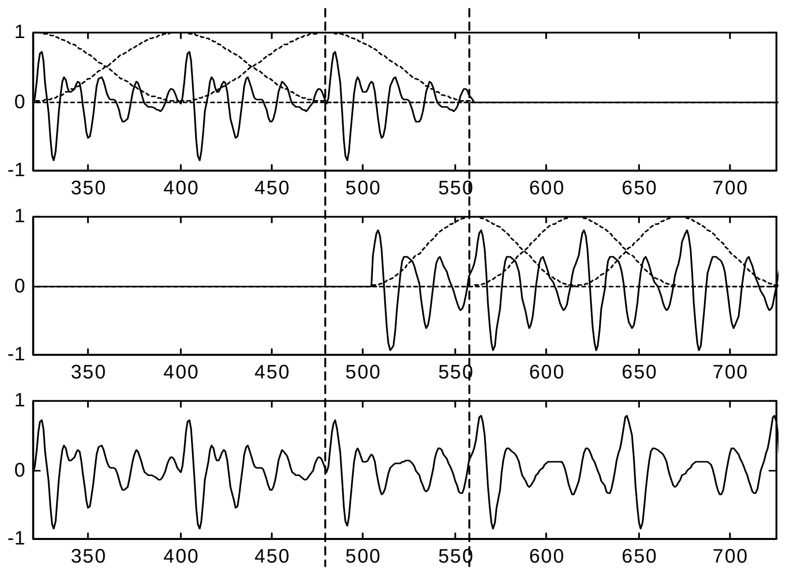

Figure 16.16 Pitch mismatches in unit concatenation. Two synthetic vowels were generated with a pitch of 138 Hz (top) and 137 Hz (middle) and exactly the same transfer function. There is no pitch tracking error, and windows are positioned coherently (no phase mismatch). The pitch of the second wave is changed through PSOLA to match the pitch of the first wave. There is a discontinuity in the resulting waveform and its spectrum (see Section 16.4.3), which is an artifact of the way the PSOLA approach estimates the spectral envelope.

In addition, pitch and timbre are not independent. Even when producing the same sound in the same phonetic context, a vastly different pitch will likely result in a different spectral envelope. This effect is particularly accentuated in the case of opera singers, who move their formants around somewhat so that the harmonics fall near the formant values and thus produce higher output.

Amplitude mismatch. A mismatch in amplitude across different units can be corrected with an appropriate amplification, but it is not straightforward to compute such a factor. More importantly, the timbre of the sound will likely change with different levels of loudness.

The PSOLA approach doesn't handle well *voiced fricatives* that are stretched considerably because of added buzziness (repeating frames induces periodicity at the high frequency that wasn't present in the original signal) or attenuation of the aspirated component (if frames are interpolated).

16.5. SOURCE-FILTER MODELS FOR PROSODY MODIFICATION

The largest problem in concatenative synthesis occurs because of spectral discontinuities at unit boundaries. The methods described in Section 16.3 significantly reduce this problem but do not eliminate it. While PSOLA can do high-quality prosody modification on speech segments, it doesn't address these spectral discontinuities occurring at unit boundaries. It would be useful to come up with a technique that allows us to smooth these spectral discontinuities. In addition, PSOLA introduces buzziness for overstretched voiced fricatives. In the following sections we describe a number of techniques that have been proposed to cope with these problems and that are based on source-filter models.

The use of source-filter models allow us to modify the source and filter separately and thus maintain more control over the resulting synthesized signal. In Section 16.5.1 we study an extension of PSOLA that allows filter modification as well for smoothing purposes. Section 16.5.2 describes mixed excitation models that also allow for improved voiced fricatives. Finally, Section 16.5.3 studies a number of voice effects that can be achieved with a source-filter model.

16.5.1. Prosody Modification of the LPC Residual

A method known as LP-PSOLA that has been proposed to allow smoothing in the spectral domain is to do PSOLA on LPC residual. This approach, thus, implicitly uses the LPC spectrum as the spectral envelope instead of the spectral envelope interpolated from the harmonics (see Section 16.4.3) when doing F0 modification. If the LPC spectrum is a better fit to the spectral envelope, this approach should reduce the spectral discontinuities due to different pitch values at the unit boundaries. LP-PSOLA reduces the *bandwidth widening*. In practice, however, this hasn't proven to offer a significant improvement in quality, possibly because the spectral discontinuities due to coarticulation dominate the overall quality.

The main advantage of this approach is that it allows us to smooth the LPC parameters around a unit boundary and thus obtain smoother speech. Since smoothing the LPC parameters directly may lead to unstable frames, other equivalent representations, such as line spectral frequencies, reflection coefficients, log-area ratios, or autocorrelation coefficients, are used instead. The use of a long window for smoothing may blur sharp spectral changes that occur in natural speech. In practice, a window of 20–50 ms centered around the boundary has been proven useful.

While time-domain PSOLA has low computational complexity, its use in a concatenative speech synthesizer generally requires a large acoustic inventory. In some applications this is unacceptable, and it needs to be compressed using any of the coding techniques described in Chapter 7. You need to keep in mind that to use such encoders you need to store the coder's memory so that the first frame of the unit can be accurately encoded. The combined decompression and prosody modification is not as computationally efficient as time-domain PSOLA alone, so that the LP-PSOLA approach may offer an effective tradeoff, given that many speech coders encode the LPC parameters anyway.

16.5.2. Mixed Excitation Models

The block diagram of PSOLA shown in Figure 16.10 for voiced sounds also works for unvoiced sounds by choosing arbitrary epochs. The time-varying filter of Figure 16.10 can be kept in its time-domain form or in the frequency domain $X_t[k]$ by taking the FFT of $x_t[n]$.

It has been empirically shown that for unvoiced frames, the phase of $X_t[k]$ is unimportant as long as it is random. Thus, we can pass random noise through a filter with magnitude response $|X_t[k]|$ and obtain perceptually indistinguishable results. This reduced representation is shown in Figure 16.17. Moreover, it has been shown that the magnitude spectrum does not need to be encoded accurately, because it doesn't affect the synthesized speech. The only potential problem with this model occurs when voiced frames are incorrectly classified as unvoiced.

Maintaining the phase of $X_t[k]$ is perceptually important for voiced sounds. If it is set to 0, two audible distortions appear: the reconstructed speech exhibits a noisy quality, and voiced fricatives sound buzzy.

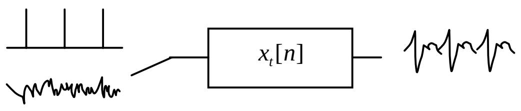

Figure 16.17 Speech synthesis model with white noise or an impulse train driving a time-varying filter.

The perceived noise may come from the fact that a listener who hears a formant, because of its amplitude spectrum, also expects the 180° phase shift associated with a complex pole. In fact, it is not the absolute phase, but the fact that if the formant frequency/bandwidth changes with time, there is a phase difference over time. If such a phase is not present, scene analysis done in the auditory system may match this to noise. This effect can be greatly attenuated if the phase of the residual in LP-PSOLA is set to 0, possibly because the LPC coefficients carry most of the needed phase information.

The buzziness in voiced fricatives is the result of setting phase coherence not only at low frequencies but also at high frequencies, where the aspiration component dominates. This is the result of treating the signal as voiced, when it has both a voiced and an unvoiced component. In fact, most voiced sounds contain some aperiodic component, particularly at high frequencies. The amount of aspiration present in a sound is called *breathiness*. Female speech tends to be more breathy than male speech [29]. Mixed-excitation models, such as those in Figure 16.18, are then proposed to more accurately represent speech.

Such a model is very similar to the waveform-interpolation coding approach of Chapter 7, and, hence, we can leverage much of what was described there regarding the estimation of $x_t^v[n]$ and $x_t^u[n]$. This approach allows us to integrate compression with prosody modification.

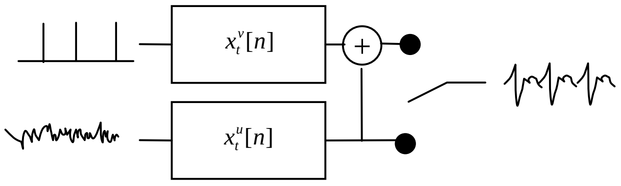

Figure 16.18 Mixed excitation model for speech synthesis.

The harmonic-plus-noise [50] model decomposes the speech signal s(t) as a sum of a random component $s_r(t)$ and a harmonic component $s_p(t)$

$$s(t) = s_r(t) + s_p(t) \tag{16.35}$$

where $s_p(t)$ uses the sinusoidal model described in Chapter 7:

$$s_p(t) = \sum_{k=1}^{K(t)} A_k(t)\cos(k\theta(t) + \phi_k(t)) \tag{16.36}$$

where $A_k(t)$ and $\phi_k(t)$ are the amplitude and phase at time t of the kth harmonic, and $\theta(t)$ is given by

$$\theta(t) = \int_{-\infty}^{t} \omega_0(l)dl \tag{16.37}$$

16.5.3. Voice Effects

One advantage of using a spectral representation like those described in this section is that several voice effects can be achieved relatively easily, such as whisper, voice conversion, and echo/reverberation.

A whispering effect can be achieved by replacing the voiced component by random noise. Since the power spectrum of the voiced signal is a combination of the vocal tract and the glottal pulse, we would need to remove the spectral roll-off of the glottal pulse. This means that the power spectrum of the noise has to be high-pass in nature. Using white noise results unnatural speech.

Voice conversion can be accomplished by altering the power spectrum [6, 28]. A warping transformation of the frequency scale can be achieved by shifting the LSF or the LPC roots, if using an LPC approach, or a warping curve if using an FFT representation.

Adding a controlled number of delayed and attenuated echoes can enhance an otherwise *dry* signal. If the delay is longer, it can simulate the room acoustics of a large hall.

16.6. EVALUATION OF TTS SYSTEMS

How do we determine whether one TTS system is better than another? Being able to evaluate TTS systems allows a customer to select the best system for his or her application. TTS evaluation is also important for developers of such systems to set some numerical goals in their design. As in any evaluation, we need to define a metric, which generally is dependent on the particular application for which the customer wants the TTS system. Such a metric consists of one or several variables of a system that are measured. Gibbon et al. [19] present a good summary of techniques used in evaluation of TTS systems.

Here we present a taxonomy of a TTS evaluation:

- *Glass-box vs. black-box evaluation*. There are two types of evaluation of TTS systems according to whether we evaluate the whole system or just one of its components: black-box and glass-box. A black-box evaluation treats the TTS system as a black box and evaluates the system in the context of a real-world application. Thus, a system may do very well on a flight reservation application but poorly on an e-mail reading application. In a glass-box evaluation, we attempt to obtain diagnostics by evaluating the different components that make up a TTS system.
- *Laboratory vs. field*. We can also conduct the study in a laboratory or in the field. While the former is generally easier to do, the latter is generally more accurate.
- *Symbolic vs. acoustic level*. In general, TTS evaluation is normally done by analyzing the output waveform, the so-called acoustic level. Glass-box evaluation at the symbolic level is useful for the text analysis and phonetic module, for example.
- *Human vs. automated*. There are two fundamentally distinct ways of evaluating speech synthesizers, according to how a given attribute of the system is estimated. One is to use human subjects; the other to automate the evaluation process. Both types have some issues in common and a number of dimensions of systematic variation. But the fundamental distinction is one of cost. In system development, and particularly in research on high-quality systems, it can be prohibitively expensive to run continuously a collection of human assessments of every algorithmic change or idea. Though human-subject checkpoints are needed throughout the development process, human testing is of greatest importance for the integrated, functionally complete system in the target field setting. At all earlier stages of development, automated testing should be substituted for human-subject testing wherever possible. The hope is that someday TTS research can be conducted as ASR research is today: algorithms are checked for accuracy or performance improvements automatically in the lab, while human subjects are mainly used when the final integrated system is deployed for field testing. This allows for rapid progress in the basic algorithms contributing to accuracy on any given dimension.

- *Judgment vs. functional testing*. Judgment tests are those that measure the TTS system in the context of the application where it is used, such as what percentage of the time users hang up an IVR system. System *A* may be more appropriate than system *B* for a banking application where most of the speech consists of numerical values, and system *B* may be better than system *A* for reading e-mail over the phone. Nonetheless, it is useful to use functional tests that measure task-independent variables of a TTS system, since such tests allow an easier comparison among different systems, albeit a nonoptimal one. Since a human listener is the consumer of a TTS system, tests have been designed to determine the following characteristics of synthesized speech: intelligibility, overall quality, naturalness, suitability for a given task, and pleasantness. In addition, testing has been used for ranking and comparing a number of competing speech synthesizers, and for comparing synthetic with natural speech.
- *Global vs. analytic assessment*. The tests can measure such *global* aspects as *overall quality*, *naturalness*, and *acceptability*. Analytic tests can measure the rate, the clarity of vowels and consonants, the appropriateness of stresses and accents, pleasantness of voice quality, and tempo. Functional tests have been designed to test the intelligibility of individual sounds (phoneme monitoring), of combinations of sounds (syllable monitoring), and of whole words (word monitoring) in isolation as well as in various types of context.

It should be noted that all the above tests focus on segments, words, and sentences. This is a historical artifact, and as the field evolves, we should see an emphasis on testing of higher-order units. The diagnostic categories mentioned above can be used as a basis for developing tests of systems that take other structure into account. Such systems might include *document-to-speech*, *concept-to-speech*, and simulated conversation or dialog. A good system will reflect document and paragraph structure in the pausing and rhythm. Concept-to-speech systems claim to bring fuller knowledge of the intended use of information to bear in message generation and synthesis. Simulated dialog systems, or human-computer dialog systems, have to mimic a more spontaneous style, which is a subtle quality to evaluate. The tricky issue with higher-order units is the difficulty of simple choice or transcription-oriented measures. To develop tests of higher-order synthesizers, the word and sentence metrics can be applied to components and the overall system until reasonable intelligibility can be verified. Then tests of the special issues raised by higher-order systems can be conducted. Appropriate measures might be MOS overall ratings, preference between systems, summarization/gist transcription with subjective scoring, and task-based measures such as following directions. With task-based testing of higher-order units, both the correctness of direction-following and the time to completion, an indirect measure of intelligibility, pleasantness, and fatigue, can be recorded.

Furthermore, speech perception is not simply auditory. As discussed in Chapter 2, the McGurk effect [36] shows that perception of a speech sound is heavily influenced by visual cues. Synthetic speech is thus perceived with higher quality when a talking head is added as a visual cue [9, 18].

Glass-box evaluation of the text analysis and phonetic analysis modules, requiring evaluation at the *symbolic* level, is done in Chapter 14. A glass-box evaluation of the prosody module is presented in Chapter 15. In this section we include glass-box evaluation of the synthesis module, as well as a black-box evaluation of the whole system.

16.6.1. Intelligibility Tests

A critical measurement of a TTS system is whether or not human listeners can understand the text read by the system. Tests that measure this are called *intelligibility tests*. In this section we describe the Diagnostic Rhyme Test, the Modified Rhyme Test, the Phonetically Balanced word list test, the Haskins Syntactic Sentence Test, and the Semantically Unpredictable Sentence Test. The first three were described in a procedure approved by the American National Standards Institute [5].

Table 16.13 The 192 stimulus words of the Diagnostic Rhyme Test (DRT).

Voicing		Nasality		Sustenation		Sibilation		Graveness		Compactness	
veal	feel	meat	beat	vee	bee	zee	thee	weed	reed	yield	wield
bean	peen	need	deed	sheet	cheat	cheep	keep	peak	teak	key	tea
gin	chin	mitt	bit	vill	bill	jilt	gilt	bid	did	hit	fit
dint	tint	nip	dip	thick	tick	sing	thing	fin	thin	gill	dill
zoo	sue	moot	boot	foo	pooh	juice	goose	moon	noon	coop	poop
dune	tune	news	dues	shoes	choose	chew	coo	pool	tool	you	rue
vole	foal	moan	bone	those	doze	joe	go	bowl	dole	ghost	boast
goat	coat	note	dote	though	dough	sole	thole	fore	thor	show	so
zed	said	mend	bend	then	den	jest	guest	met	net	keg	peg
dense	tense	neck	deck	fence	pence	chair	care	pent	tent	yen	wren
vast	fast	mad	bad	than	dan	jab	gab	bank	dank	gat	bat
gaff	calf	nab	dab	shad	chad	sank	thank	fad	thad	shag	sag
vault	fault	moss	boss	thong	tong	jaws	gauze	fought	thought	yawl	wall
daunt	taunt	gnaw	daw	shaw	chaw	saw	thaw	bong	dong	caught	thought
jock	chock	mom	bomb	von	bon	jot	got	wad	rod	hop	fop
bond	pond	knock	dock	vox	box	chop	cop	pot	tot	got	dot

Among the best known and most mature of these tests is the Diagnostic Rhyme Test (DRT) proposed by Voiers [54], which provides for diagnostic and comparative evaluation of the intelligibility of single initial consonants. The test runs twice through the list of 96 rhyming pairs shown in Table 16.13. The test consists of identification choice between two alternative English (or target-language) words, differing by a single phonetic feature in the initial consonant. For English the test includes contrasts among easily confusable paired consonant sounds such as *veal/feel, meat/beat, fence/pence, cheep/keep, weed/reed*, and *hit/fit*. In the test, both *veal* and *feel* are presented with the response alternatives *veal* and *feel*. Six contrasts are represented, namely voicing, nasality, sustention, sibilation, graveness, and compactness. Each contrast is included 32 times in the test, combined with 8 different vowels. The percentage of right answers is used as an indicator of speech synthesizer intelligibility. The tests use a minimum of five talkers and five listeners; larger subject groups reduce the margin of error. Even for high-quality speech coders, 100% *correct* responses are rarely achieved, so synthesizer results should be interpreted generously.

Table 16.14 The 300 stimulus words of the Modified Rhyme Test (MRT).

went	sent	bent	dent	tent	rent	same	name	game	tame	came	fame
hold	cold	told	fold	sold	gold	peel	reel	feel	eel	keel	heel
pat	pad	pan	path	pack	pass	hark	dark	mark	bark	park	lark
lane	lay	late	lake	lace	lame	heave	hear	heat	heal	heap	heath
kit	bit	fit	hit	wit	sit	cup	cut	cud	cuff	cuss	cud
must	bust	gust	rust	dust	just	thaw	law	raw	paw	jaw	saw
teak	team	teal	teach	tear	tease	pen	hen	men	then	den	ten
din	dill	dim	dig	dip	did	puff	puck	pub	pus	pup	pun
bed	led	fed	red	wed	shed	bean	beach	beat	beak	bead	beam
pin	sin	tin	fin	din	win	heat	neat	feat	seat	meat	beat
dug	dung	duck	dud	dub	dun	dip	sip	hip	tip	lip	rip
sum	sun	sung	sup	sub	sud	kill	kin	kit	kick	king	kid
seep	seen	seethe	seek	seem	seed	hang	sang	bang	rang	fang	gang
not	tot	got	pot	hot	lot	took	cook	look	hook	shook	book
vest	test	rest	best	west	nest	mass	math	map	mat	man	mad
pig	pill	pin	pip	pit	pick	ray	raze	rate	rave	rake	race
back	bath	bad	bass	bat	ban	save	same	sale	sane	sake	safe
way	may	say	pay	day	gay	fill	kill	will	hill	till	bill
pig	big	dig	wig	rig	fig	sill	sick	sip	sing	sit	sin
pale	pace	page	pane	pay	pave	bale	gale	sale	tale	pale	male
cane	case	cape	cake	came	cave	wick	sick	kick	lick	pick	tick
shop	mop	cop	top	hop	pop	peace	peas	peak	peach	peat	peal
coil	oil	soil	toil	boil	foil	bun	bus	but	bug	buck	buff
tan	tang	tap	tack	tam	tab	sag	sat	sass	sack	sad	sap
fit	fib	fizz	fill	fig	fin	fun	sun	bun	gun	run	nun

A variant of this is the Modified Rhyme Test (MRT) proposed by House [22], which also uses a 300-entry word list for subjective intelligibility testing. The modified Rhyme Test (shown in Table 16.14) uses 50 six-word lists of rhyming or similar-sounding monosyllabic English words, e.g., *went*, *sent*, *bent*, *dent*, *tent*, *rent*. Each word is basically Consonant-Vowel-Consonant (with a few consonant clusters), and the six words in each list differ only in the initial or final consonant sound(s). Listeners are asked to identify which of the words was spoken by the synthesizer (closed response), or in some cases to enter any word they thought they heard (open response). A carrier sentence, such as "Would you write <test word> now," is usually used for greater naturalness in stimulus presentation. Listener responses can be scored as the number of words heard correctly; or the frequency of confusions of particular consonant sounds. This can be viewed as *intelligibility* of the synthesizer.

Though this is a nice isolation of one property, and as such is particularly appropriate for diagnostic use, it is not intended to substitute for fuller evaluation under more realistic listening conditions involving whole sentences. Segmental intelligibility is somewhat overestimated in these tests, because all the alternatives are real words and the subjects can adjust their perception to match the closest word. A typical human voice gives an MRT score of about 99%, with that of TTS systems generally ranging from 70% to 95%.

The set of twenty phonetically balanced (PB) word lists was developed during World War II and has been used very widely since then in subjective intelligibility testing. In Table 16.15 we include the first four PB word lists [20]. The words in each list are presented in a new, random order each time the list is used, each spoken in the same carrier sentence. The PB intelligibility test requires more training of listeners and talkers than other subjective tests and is particularly sensitive to SNR: a relatively small change causes a large change in the intelligibility score.

Tests using the *Haskins Syntactic Sentences* [40] go somewhat farther toward more realistic and holistic stimuli. This test set consists of 100 semantically unpredictable sentences of the form *The <Adjective> <Noun1> <Verb> the <Noun2>*, such as *"The old farm cost the blood,"* using high-frequency words. Compared with the rhyme tests, contextual predictability based on meaning is largely lacking, the longer speech streams are more realistic, and more coarticulation is present. Intelligibility is indicated by percentage of words correct.

Table 16.15 Phonetically balanced word lists.

List 1	are, bad, bar, bask, box, cane, cleanse, clove, crash, creed, death, deed, dike, dish, end, feast, fern, folk, ford, fraud, fuss, grove, heap, hid, hive, hunt, is, mange, no, nook, not, pan, pants, pest, pile, plush, rag, rat, ride, rise, rub, slip, smile, strife, such, then, there, toe, use, wheat
List 2	awe, bait, bean, blush, bought, bounce, bud, charge, cloud, corpse, dab, earl, else, fate, five, frog, gill, gloss, hire, hit, hock, job, log, moose, mute, nab, need, niece, nut, our, perk, pick, pit, quart, rap, rib, scythe, shoe, sludge, snuff, start, suck, tan, tang, them, trash, vamp, vast, ways, wish
List 3	ache, air, bald, barb, bead, cape, cast, check, class, crave, crime, deck, dig, dill, drop, fame, far, fig, flush, gnaw, hurl, jam, law, leave, lush, muck, neck, nest, oak, path, please, pulse, rate, rouse, shout, sit, size, sob, sped, stag, take, thrash, toil, trip, turf, vow, wedge, wharf, who, why
List 4	bath, beast, bee, blonde, budge, bus, bush, cloak, course, court, dodge, dupe, earn, eel, fin, float, frown, hatch, heed, hiss, hot, how, kite, merge, lush, neat, new, oils, or, peck, pert, pinch, pod, race, rack, rave, raw, rut, sage, scab, shed, shin, sketch, slap, sour, starve, strap, test, tick, touch

Another test minimizing predictability is *Semantically Unpredictable Sentences* [23], with test sets for Dutch, English, French, German, Italian, and Swedish. A short template of syntactic categories provides a frame, into which words are randomly slotted from the lexicon. For example, the template <*Subject*> <*Verb*> <*Adverbial*> might appear as "*The chair ate through the green honesty.*" Fifty sentences (10 per syntactic template) are considered adequate to test a synthesizer. Open transcription is requested, and *sentences correct* is used to score a synthesizer's intelligibility. Other such tests exist, and some include systematic variation of prosody on particular words or phrases as well.

The Harvard Psychoacoustic Sentences [16] is a set of 100 meaningful, syntactically varied, phonetically balanced sentences, such as "*Add salt before you fry the egg*," requiring an o*pen response identification,* instead of a multiple-choice test.

16.6.2. Overall Quality Tests

While a TTS system has to be intelligible, this does not guarantee user acceptance, because its quality may be far from that of a human speaker. In this section we describe the Mean Opinion Score and the Absolute Category Ratings.

Human-subject judgment testing for TTS can adapt methods from speech-coding evaluation (see Chapter 7). With speech coders, *Mean Opinion Score* (MOS) is administered by asking 10 to 30 listeners to rate several sentences of coded speech on a scale of 1 to 5 (1 = Bad, 2 = Poor, 3 = Fair, 4 = Good, 5 = Excellent). The scores are averaged, resulting in an overall MOS rating for the coder. This kind of methodology can be applied to speech synthesizers as well. Of course, as with any human subject test, it is essential to carefully design the listening situation and carefully select the subject population, controlling for education, experience, physiological disorders, dialect, etc. As with any statistically interpreted test, the standard analyses of score distributions, standard deviation, and confidence intervals must be performed. The range of quality in coder evaluations by MOS are shown in Table 16.16.

Table 16.16 Mean opinion score (MOS) ratings and typical interpretations.

MOS Scores	Quality	Comments
4.0–4.5	Toll/Network	Near-transparent, "in-person" quality
3.5–4.0	Communications	Natural, highly intelligible, adequate for telecommunications, changes and degradation of quality very noticeable
2.5–3.5	Synthetic	Usually intelligible, can be unnatural, loss of speaker recognizability, inadequate levels of naturalness

Since we are making the analogy to coders, certain ironies can be noted. Note the lowest-range descriptor for coder evaluation: *synthetic*. In using MOS for synthesis testing, output is being evaluated by implicit reference to real human speech, and the upper range in the coder MOS interpretations above (3.5–4.5) is probably not applicable to the output of most TTS systems. Even a *good* TTS system might fare poorly on such a coder MOS evaluation. Therefore, the MOS interpretive scale, when applied to synthesis, cannot be absolute as the above coding-based interpretive table would imply. Furthermore, subjects participating in MOS-like tests of synthesizers should be made aware of the special nature of the speech (synthetic) and adjust their expectations accordingly. Finally, no matter how carefully the test is designed and administered, it is difficult to correlate, compare, and scale such measures. Nevertheless, MOS tests are perhaps suited to relative ranking of various synthesizers. The 1-to-5 scale is categorical, but similar judgment tests can be run in *magnitude* mode, with the strength of the quality judgment being indicated along a continuous scale, such as a moving slider bar.

Table 16.17 Listening Quality Scale.

Quality of the Speech	Score
Excellent	5
Good	4
Fair	3
Poor	2
Bad	1

The International Telecommunication Union (ITU) has attempted to specify some standards for assessing synthetic speech, including spliced digitized words and phrases, typically with the expectation of delivery over the phone. The *Absolute Category Rating* (ACR) system recommended by ITU P.800 offers instructions to be given to subjects for making category judgments in MOS-style tests of the type discussed here. The first is the *Listening Quality Scale*, shown in Table 16.17, and the second the *Listening Effort Scale* shown in Table 16.18.

Table 16.18 Listening Effort Scale.

Effort Required to Understand the Meanings of Sentences	Score
Complete relaxation possible; no effort required	5
Attention necessary; no appreciable effort required	4
Moderate effort required	3
Considerable effort required	2
No meaning understood with any feasible effort	1

It is sometimes possible to get subjects to pay particular attention to various particular features of the utterance, which may be called *analytic* as opposed to *global* listening. The desired features generally have to be described somehow, and these descriptions can be a bit vague. Thus, standard measures of reliability and validity, as well as result normalization, must be applied. Typical descriptors for important factors in analytic listening might be: *smoothness*, *naturalness*, *pleasantness*, *clarity*, *appropriateness*, etc., each tied to a particular underlying target quality identified by the system designers. For example, smoothness might be a descriptor used when new algorithms for segment concatenation and blending are being evaluated in a concatenative system. Naturalness might be the quality descriptor when a formant-based system has been made more natural by incorporation of a richer glottal source function. Some elements of the speech can be more directly identified to the subject in familiar terms. For example, Pleasantness might be a way of targeting the pitch contours for attention, or the subject could be specifically asked to rank the pitch contours per se, in terms of naturalness, pleasantness, etc. Appropriateness might be a way of getting at judgments of accentuation: e.g., a stimulus that was accented as "… *birthday PARTY*" might be judged less appropriate, in a neutral semantic context, than one that was perceived as "… *BIRTHDAY party*." But no matter how the attributes are described, in human-subject MOS-style testing there cannot be a clear and consistent separation of effects.

16.6.3. Preference Tests

Normalized MOS scores for different TTS systems can be obtained without any direct preference judgments. If direct comparisons are desired, especially for systems that are informally judged to be fairly close in quality, another ITU recommendation, the *Comparison Category Rating* (CCR) method, may be used. In this method, listeners are presented with a pair of speech samples on each trial. The order of the *system A system B* samples is chosen at random for each trial. On half of the trials, the *system A* sample is followed by the *system B* sample. On the remaining trials, the order is reversed. Listeners use the instructions in Table 16.19 to judge the quality of the second sample relative to that of the first. Sometimes the granularity can be reduced as much as simply "*prefer A/prefer B*."

Assuming (A,B) is the reference presentation order, scores for the (B,A) presentations may be normalized by reversing their signs (e.g., –1 in B,A order becomes 1, etc.). Subsequently, standard statistical summarizations may be performed, like the one described in Chapter 3.

Table 16.19 Preference ratings between two systems. The quality of the second utterance is compared to the quality of the first by means of 7 categories. Sometimes only better, same, or worse are used.

3	Much Better
2	Better
1	Slightly Better
0	About the Same
-1	Slightly Worse
-2	Worse
-3	Much Worse

16.6.4. Functional Tests

Functional testing places the human subject in the position of carrying out some task related to, or triggered by, the speech. This can simulate a full field deployment, with a usercentric task, or can be more of a laboratory situation, with a testcentric task. In the laboratory situation, various kinds of tasks have been proposed. In *analytic* mode, functional testing can enforce isolation of the features to be attended to in the structure of the test stimuli themselves. This can lead to a more precise form of result than the MOS judgment approach. There have been a wide variety of proposals and experiments of this type.

One of the well-known facts in TTS evaluation is that the quality of a system is dominated by the quality of its worst component. While it may be argued that it is impossible to separate the effects of the front-end analysis and back-end synthesis, it is convenient to do so to gain a better understanding of each component. An attempt to study the quality of the speech synthesis module has been done via the use of natural instead of synthetic prosody. This way, it is presumed that the prosody module is doing the best possible job, and that any problem is then due to a deficient speech synthesis. The natural pitch contour can be obtained with a pitch tracker (or using a laryngograph signal), and the natural durations can be obtained either through manual segmentation or through the use of a speech recognition system used in forced-alignment mode. Plumpe and Meredith [44] conducted a preference test between original recordings and waveforms created when one of the modules of a concatenative TTS system used synthetically generated values instead of the natural values. The results indicated that using synthetic pitch instead of natural pitch was the cause of largest degradation according to listeners, and, thus, that pitch generation was the largest bottleneck in the system. The pitch-generation module was followed by the spectral discontinuities at the concatenation points, with duration being the least damaging.

Some functional tests are much more creative than simple transcription, however. They could, in theory, border on related areas, such as memory testing, involving summarizing passages, or following synthesized directions, such as a route on a map. The ultimate test of synthesis, in conjunction with all other language interface components, is said to be the Turing test [53]. In this amusing scenario, a human being is placed into conversation with a computational agent, represented vocally for our purposes, perhaps over the telephone. As Turing put it: “It is proposed that a machine may be deemed intelligent, if it can act in such a manner that a human cannot distinguish the machine from another human merely by asking questions via a mechanical link.” Turing predicted that in the future “an average interrogator will not have more than a 70 percent chance of making the right identification, human or computer on the other end, after five minutes of questioning” in this game. A little reflection might raise objections to this procedure as a check on speech output quality per se, since some highly intelligent people have speech disabilities, but the basic idea should be clear, and it remains an amusing Holy Grail for the artificial intelligence field generally. Of course, no automated or laboratory test can substitute for a real-world trial with paying customers.

16.6.5. Automated Tests

The tests described above always involved the use of human subjects and are the best tests that can be used to evaluate a TTS system. Unfortunately, they are time consuming and expensive to conduct. This limits their application to an infrequent use, which can hardly have any diagnostic value. Automated *objective* tests usually involve establishing a test corpus of correctly tagged examples of the tested phenomena, which can be automatically checked. This style of testing is particularly appropriate when working with isolated components of the TTS system, for diagnosis or regression testing (glass-box testing). It is not particularly productive to discuss such testing in the abstract, as the test features must closely track each system’s design and implementation. Nevertheless, a few typical areas for testing can be noted. In general, tests are simultaneously testing the linguistic model and content as well as the software implementation of a system, so whenever a discrepancy arises, both possible sources of error must be considered.

Several automated tests for text analysis and letter-to-sound conversion are presented in Chapter 14. A number of automated tests for prosody are discussed in Chapter 15. Here we touch on automated tests for the synthesis module.

The ITU has also created the P.861 proposal for estimating perceptual scores using automated signal-based measurements. The P.861 specifies a particular technique known as *Perceptual Speech Quality Measurement* (PSQM). In this method, for each analysis frame, various quantified measures based on the time signal, the power spectrum, the Bark power spectrum, the excitation spectrum, the compressed loudness spectrum, etc. of both the reference and the test signal can be computed. In some cases the PSQM score can be converted to an estimated MOS score, with interpretations similar to those of Table 16.16. At present such methods are limited primarily to analysis of telephone-quality speech (300–3400 Hz bandwidth), to be compared with closely related reference utterances. This method could perhaps be adapted to stand in for human judgments during system development of new versions of modules, say glottal source functions in a formant synthesizer, comparing the resulting synthetic speech to a standard reference system's output on a given test sample.

16.7. HISTORICAL PERSPECTIVE AND FUTURE READING[3]

In 1779 in St. Petersburg, Russian Professor Christian Kratzenstein explained physiological differences between five long vowels (/*a*/, /*e*/, /*i*/, /*o*/, and /*u*/) and made apparatus to produce them artificially. He constructed acoustic resonators similar to the human vocal tract and activated the resonators with vibrating reeds as in music instruments. Von Kempelen (1734–1804) proposed in 1791 in Vienna a mechanical speaking machine that could produce not just vowels but whole words and sentences (see Figure 16.19). While working with his speaking machine, he demonstrated a speaking chess-playing machine. Unfortunately, the main mechanism of the machine was a concealed, legless chess-player expert. Therefore, his real speaking machine was not taken as seriously as it should have been. In 1846, Joseph Faber developed a synthesizer, called speech organ, that had more control of pitch to the extent it could sing *God Save the Queen* in a performance in London.

The first electrical synthesis device was introduced by Stewart in 1922 [4]. The device had a buzzer as excitation and two resonant circuits to model the acoustic resonances of the vocal tract and was able to generate single static vowel sounds with the first two formants. In 1932 Japanese researchers Obata and Teshima added a third formant for more intelligible vowels.

Homer Dudley of Bell Laboratories demonstrated at the 1939 New York World's Fair the *Voder*, the first electrical speech synthesizer, which was human-controlled. The operator worked at a keyboard, with a wrist bar to control the voicing parameter and a pedal for pitch control (see Figure 16.20 and Figure 16.21), and it was able to synthesize continuous

[3] Chapter 6 includes a historical prospective on representation of speech signals that is intimately tied to speech synthesis.

speech. The *Pattern Playback* is an early talking machine that was built by Franklin S. Cooper and his colleagues at Haskins Laboratories in the late 1940s. This device synthesized sound by passing light through spectrograms that in turn modulated an oscillator with a fixed F0 of 120 Hz and 50 harmonics.

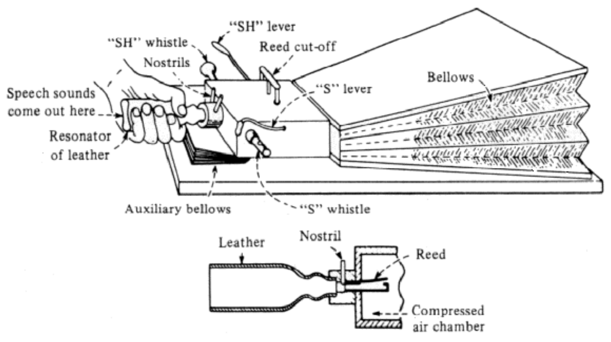

Figure 16.19 Wheatstone's reconstruction of von Kempelen's speaking machine [14] (after Flanagan [17]).

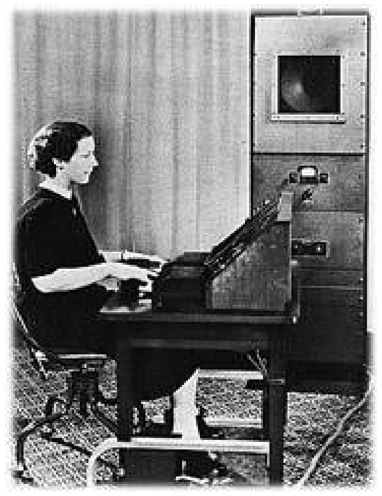

Figure 16.20 The Voder developed by Homer Dudley of Bell Labs at the 1939 World's Fair in New York. The operator worked at a keyboard, with a wrist bar to control the voicing parameter and a pedal for pitch control.

The first analog parallel formant synthesizer, the *Parametric Artificial Talker* (PAT), was developed in 1953 by Walter Lawrence of the Signals Research and Development Establishment of the British Government. Gunnar Fant of the KTH in Sweden developed an analog cascade formant synthesizer, the OVE II. Both Lawrence and Fant showed in 1962 that by manually tuning the parameters, a natural sentence could be reproduced reasonably faithfully. Acoustic analog synthesizers were also known as terminal analogs, resonance-synthesizers. John Holmes tuned by hand the parameters of his formant synthesizer so well that the average listener could not tell the difference between the synthesized sentence "*I enjoy the simple life*" and the natural one [31].

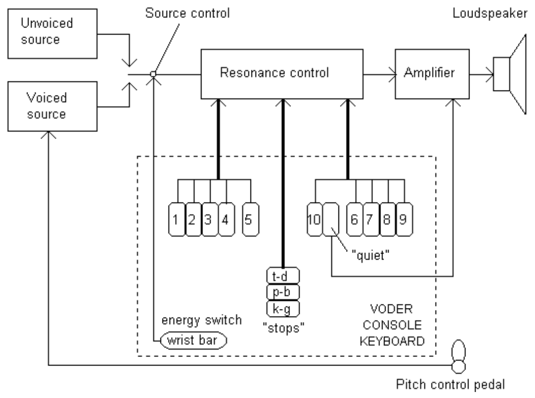

Figure 16.21 Block diagram of the Voder by Homer Dudley, 1939 (after Flanagan [17]).

The first articulatory synthesizer, the DAVO, was developed in 1958 by George Rosen at M.I.T. Cecil Coker designed rules to control a low-dimensionality articulatory model in 1968. Paul Mermelstein and James Flanagan from Bell Labs also used articulatory synthesis in 1976. Articulatory synthesis, however, never took off, because formant synthesis was better understood at the time.

The advent of the digital computer prompted John Kelly and Louis Gerstman to create in 1961 the first phonemic-synthesis-by-rule program. John Holmes and his colleagues Ignatius Mattingly and John Shearme developed a rule program for a formant synthesizer at JSRU in England. The first full text-to-speech system was developed by Noriko Umeda in 1968 at the Electrotechnical Laboratory of Japan. It was based on an articulatory model and included a syntactic analysis module with sophisticated heuristics. The speech was quite intelligible, but monotonous and far from the quality of present systems.

In 1976, Raymond Kurzweil developed a unique reading machine for the blind, a computer-based device that read printed pages aloud. It was an 80-pound device that shot a beam of light across each printed page, converting the reflected light into digital data that was transformed by a computer into synthetic speech. It made reading of all printed material possible for blind people, whose reading has previously been limited to material translated into Braille. The work of Dennis Klatt of MIT had a large influence in the field. In 1979 together with Jonathan Allen and Sheri Hunnicut he developed the MITalk system. Two years later Klatt introduced his famous Klattalk system, which used a new sophisticated voicing source.

The early 1980s marked the beginning of commercial TTS systems. The Klattalk system was the basis of Telesensory Systems' Prose-2000 commercial TTS system in 1982. It also formed the basis for Digital Equipment Corporation's DECtalk commercial system in 1983, probably the most widely used TTS system of the twentieth century. The Infovox TTS system, the first multilanguage formant synthesizer, was developed in Sweden by Rolf Carlson, Bjorn Granstrom, and Sheri Hunnicutt in 1982, and it was a descendant of Gunnar Fant's OVE system. The first integrated circuit for speech synthesis was probably the Votrax chip, which consisted of cascade formant synthesizer and simple low-pass smoothing circuits. In 1978 Richard Gagnon introduced an inexpensive Votrax-based Type-n-Talk system. The first work in concatenative speech synthesis was done in 1968 by Rex Dixon and David Maxey, where diphones were parametrized with formant frequencies and then concatenated. In 1977, Joe Olive and his colleagues at Bell Labs [41] concatenated linear-prediction diphones. In 1982 Street Electronics introduced the Echo system, a diphone concatenation synthesizer which was based on a newer version of the same chip as in the Speak-n-Spell toy introduced by Texas Instruments in 1980.

Concatenative systems started to gain momentum in 1985 with the development of the PSOLA prosody modification technique by France Telecom's Charpentier and Moulines. PSOLA increased the text coverage of concatenative systems by allowing diphones to have their prosody modified. The hybrid Harmonic/Stochastic (H/S) model of Abrantes [1] has also been successfully used for prosody modification. The foundation of corpus-based concatenative systems was developed by a team of researchers at ATR in Japan in the early 1990s [10, 27]. The use of a large database of long units was also pioneered by researchers at AcuVoice Inc. Other corpus-based systems have made use of HMMs to automatically segment speech databases, as well as to serve as units in concatenative synthesis [13, 24]. A significant milestone for concatenative TTS is that Microsoft integrated it [24] for the mass market in Windows 2000.

For more detailed description of speech synthesis development and history see, for example, [31] and [17] and references in these. A number of audio clips are available in Klatt [31] showing the progress through the early years. You can hear samples at the Smithsonian's Speech Synthesis History Project [35]. A Web site with comparison of recent TTS systems can be found at [33].

REFERENCES

[1] Abrantes, A.J., J.S. Marques, and I.M. Trancoso, "Hybrid Sinusoidal Modeling of Speech without Voicing Decision," *Proc. Eurospeech*, 1991, Genoa, Italy pp. 231-234.

[2] Acero, A., "Source-Filter Models for Time-Scale Pitch-Scale Modification of Speech," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1998, Seattle pp. 881-884.

[3] Acero, A., "Formant Analysis and Synthesis Using Hidden Markov Models," *Eurospeech*, 1999, Budapest pp. 1047-1050.

[4] Allen, J., M.S. Hunnicutt, and D.H. Klatt, *From Text to Speech: the MITalk System*, 1987, Cambridge, UK, University Press.

[5] ANSI, *Method for Measuring the Intelligibility of Speech Over Communication Systems*, 1989, American National Standards Institute.

[6] Arslan, L.M. and D. Talkin, "Speaker Transformation Using Sentence HMM Based Alignments and Detailed Prosody Modification," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1998, Seattle pp. 289-292.

[7] Beutnagel, M., *et al.*, "The AT&T Next-Gen TTS System," *Joint Meeting of ASA*, 1999, Berlin pp. 15-19.

[8] Bickley, C.A., K.N. Stevens, and D.R. Williams, "A Framework for Synthesis of Segments Based on Pseudoarticulatory Parameters" in *Progress in Speech Synthesis* 1997, New York, pp. 211-220, Springer-Verlag.

[9] Bregler, C., M. Covell, and M. Slaney, "Video Rewrite: Driving Visual Speech with Audio," *ACM Siggraph*, 1997, Los Angeles pp. 353-360.

[10] Campbell, W.N. and A.W. Black, "Prosody and the Selection of Source Units for Concatenative Synthesis" in *Progress in Speech Synthesis*, J.V. Santen, *et al.*, eds. 1996, pp. 279-292, Springer Verlag.

[11] Covell, M., M. Withgott, and M. Slaney, "Mach1: Nouniform Time-Scale Modification of Speech," *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1998, Seattle pp. 349-352.

[12] Crochiere, R., "A Weighted Overlap-Add Method of Short Time Fourier Analysis/Synthesis," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1980, **28**(2), pp. 99-102.

[13] Donovan, R. and P. Woodland, "Improvements in an HMM-based Speech Synthesizer," *Proc. of the EuroSpeech Conf.*, 1995, Madrid pp. 573-576.

[14] Dudley, H. and T.H. Tarnoczy, "The Speaking Machine of Wolfgang von Kempelen," *Journal of the Acoustical Society of America*, 1950, **22**, pp. 151-166.

[15] Dutoit, T., *An Introduction to Text-to-Speech Synthesis*, 1997, Kluwer Academic Publishers.

[16] Egan, J., "Articulation Testing Methods," *Laryngoscope*, 1948, **58**, pp. 955-991.

[17] Flanagan, J., *Speech Analysis Synthesis and Perception*, 1972, New York, Springer-Verlag.

[18] Galanes, F.G., *et al.*, "Generation of Lip-Synched Synthetic Faces from Phonetically Clustered Face Movement Data," *AVSP*, 1998, Terrigal, Australia.

[19] Gibbon, D., R. Moore, and R. Winski, *Handbook of Standards and Resources for Spoken Language Systems*, 1997, Berlin & New York, Walter de Gruyter Publishers.

[20] Goldstein, M., "Classification of Methods Used for Assesment of Text-to-Speech Systems According to the Demands Placed on the Listener," *Speech Communication*, 1995, **16**, pp. 225-244.

[21] Hon, H., *et al.*, "Automatic Generation of Synthesis Units for Trainable Text-to-Speech Systems," *Int. Conf. on Acoustics, Signal and Speech Processing*, 1998, Seattle pp. 293-296.

[22] House, A., *et al.*, "Articulation Testing Methods: Consonantal Differentiation with a Closed Response Set," *Journal of the Acoustical Society of America*, 1965, **37**, pp. 158-166.

[23] Howard-Jones, P., *SOAP, Speech Output Assessment Package*, , 1992, ESPRIT SAM-UCL-042.

[24] Huang, X., *et al.*, "Whistler: A Trainable Text-to-Speech System," *Int. Conf. on Spoken Language Processing*, 1996, Philadephia pp. 2387-2390.

[25] Huang, X., *et al.*, "Recent Improvements on Microsoft's Trainable Text-To-Speech System - Whistler," *Int. Conf. on Acoustics, Signal and Speech Processing*, 1997, Munich, Germany pp. 959-962.

[26] Hunt, A.J. and A.W. Black, "Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database," *Int. Conf. on Acoustics, Speech and Signal processing*, 1996, Atlanta pp. 373-376.

[27] Iwahashi, N., N. Kaiki, and Y. Sagisaka, "Concatenation Speech Synthesis by Minimum Distortion Criteria," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1992, San Francisco pp. 65-68.

[28] Kain, A. and M. Macon, "Text-to-Speech Voice Adaptation from Sparse Training Data," *Int. Conf. on Spoken Language Systems*, 1998, Sydney, Australia pp. 2847-2850.

[29] Klatt, D. and L. Klatt, "Analysis, Synthesis and Perception of Voice Quality Variations among Female and Male Talkers," *Journal of the Acoustical Society of America*, 1990, **87**, pp. 737-793.
[30] Klatt, D.H., "Software for a Cascade/Parallel Formant Synthesizer," *Journal of Acoustical Society of America*, 1980, **67**, pp. 971-995.
[31] Klatt, D.H., "Review of Text to Speech Conversion for English," *Journal of the Acoustical Society of America*, 1987, **82**, pp. 737-793.
[32] Krishnamurthy, A.K. and D.G. Childers, "Two Channel Speech Analysis," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1986, **34**, pp. 730-743.
[33] LDC, *Interactive Speech Synthesizer Comparison Site*, 2000, http://morph.ldc.upenn.edu.
[34] Maachi, M., "Coverage of Names," *Journal of the Acoustical Society of America*, 1993, **94**(3), pp. 1842.
[35] Maxey, H., *Smithsonian Speech Synthesis History Project*, 2000, http://www.mindspring.com/~dmaxey/ssshp/.
[36] McGurk, H. and J. MacDonald, "Hearing Lips and Seeing Voices," *Nature*, 1976, **264**, pp. 746-748.
[37] Mermelstein, P. and M.R. Schroeder, "Determination of Smoothed Cross-Sectional Area Functions of the Vocal Tract from Formant Frequencies," *Fifth Int. Congress on Acoustics*, 1965.
[38] Moulines, E. and F. Charpentier, "Pitch-Synchronous Waveform Processing Techniques for Text-to-Speech Synthesis Using Diphones," *Speech Communication*, 1990, **9**(5), pp. 453-467.
[39] Moulines, E. and W. Verhelst, "Prosodic Modifications of Speech" in *Speech Coding and Synthesis*, W.B. Kleijn and K.K. Paliwal, eds. 1995, pp. 519-555, Elsevier.
[40] Nye, P. and J. Gaitenby, *The Intelligibility of Synthetic Monosyllabic Words in Short, Syntactically Normal Sentences*, 1974, Haskins Laboratories.
[41] Olive, J., "Rule Synthesis of Speech from Dyadic Units," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1977, Hartford, CT pp. 568-570.
[42] Olive, J.P., A. Greenwood, and J.S. Coleman, *Acoustics of American English Speech: a Dynamic Approach*, 1993, New York, Springer-Verlag.
[43] Oliveira, L., "Estimation of Source Parameters by Frequency Analysis," *Proc. of the Eurospeech Conf.*, 1993, Berlin pp. 99-102.
[44] Plumpe, M. and S. Meredith, "Which is More Important in a Concatenative Text-to-Speech System: Pitch, Duration, or Spectral Discontinuity," *Third ESCA/COCOSDA Int. Workshop on Speech Synthesis*, 1998, Jenolan Caves, Australia pp. 231-235.
[45] Roucos, S. and A. Wilgus, "High Quality Time-Scale Modification of Speech," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1985 pp. 493-496.
[46] Sagisaka, Y. and N. Iwahashi, "Objective Optimization in Algorithms for Text-to-Speech Synthesis" in *Speech Coding and Synthesis*, W.B. Kleijn and K.K. Paliwal, eds. 1995, pp. 685-706, Elsevier.
[47] Santen, J.V., "Combinatorial Issue in Text-to-Speech Synthesis," *Proc. of the Eurospeech Conf.*, 1997, Rhodes, Greece pp. 2511-2514.
[48] Sproat, R., *Multilingual Text-To-Speech Synthesis: The Bell Labs Approach*, 1998, Dordrecht, Kluwer Academic Publishers.
[49] Stylianou, Y., "Assessment and Correction of Voice Quality Variabilities in Large Speech Databases for Concatenative Speech Synthesis," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1999, Phoenix, AZ pp. 377-380.
[50] Stylianou, Y., J. Laroche, and E. Moulines, "High Quality Speech Modification based on a Harmonic+ Noise mode," *Proc Eurospeech*, 1995, Madrid.

[51] Syrdal, A., A. Conkie, and Y. Stylianou, "Exploration of Acoustic Correlates in Speaker Selection for Concatenative Synthesis," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia pp. 2743-2746.

[52] Syrdal, A., *et al.*, "Voice Selection for Speech Synthesis," *Journal of the Acoustical Society of America*, 1997, **102**, pp. 3191.

[53] Turing, A.M., "Computing Machinery and Intelligence," *Mind*, 1950, **LIX**(236), pp. 433-460.

[54] Voiers, W., A. Sharpley, and C. Hehmsoth, *Research on Diagnostic Evaluation of Speech Intelligibility*, 1975, Air Force Cambridge Research Laboratories, Bedford, MA.

[55] Yi, J., *Natural Sounding Speech Synthesis Using Variable-Length Units*, Masters' Thesis in *EECS Dept.* 1998, MIT, Cambridge, MA.